

Это можно не проверять, оно в отдельном доке

Учреждение образования  
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет информационных технологий и управления

Кафедра информационных технологий автоматизированных систем

*К защите допустить:*

Заведующий кафедрой ИТАС

А.А. Навроцкий

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к дипломному проекту

на тему

АВТОМАТИЗИРОВАННАЯ СИСТЕМА УПРАВЛЕНИЯ ПОЛОЧНЫМ  
ПРОСТРАНСТВОМ СУПЕРМАРКЕТА

БГУИР ДП 1-53 01 02 06 072 ПЗ

Студент

В. Г. Колесников

Руководитель

В. Л. Арановский

Консультанты:

*от кафедры ИТАС*

Т. С. Боброва

*по экономической части*

Т. Л. Слюсарь

Нормоконтролер

В. И. Ярмолик

Рецензент

Минск 2020

## РЕФЕРАТ

АВТОМАТИЗИРОВАННАЯ СИСТЕМА УПРАВЛЕНИЯ ПОЛОЧНЫМ ПРОСТРАНСТВОМ СУПЕРМАРКЕТА: дипломный проект / В. Г. Колесников. – Минск : БГУИР, 2020, – п.з. – 85 с., чертежей (плакатов) – 6 л. формата А1.

Цель работы: разработка автоматизированной системы управления полочным пространством супермаркета.

Пояснительная записка к дипломному проекту состоит из введения, 4 разделов, включающих характеристику объекта разработки, разработку автоматизированной системы управления полочным пространством супермаркета, программную реализацию автоматизированной системы, а также технико-экономическое обоснование разработки и использования системы, заключение, список использованных источников.

Для разработки программного комплекса был выбран язык программирования *Python*. Система представляет собой клиентское приложение, написанное с использованием *HTML*, *CSS* и *JavaScript* с использованием технологии *AJAX*, и серверное приложение, написанное на фреймворке *Django*. Доступ к приложению пользователи будут получать посредством веб-браузера.

Результатом работы над дипломным проектом стала спроектированная и разработанная автоматизированная система управления полочным пространством супермаркета, подготовленное руководство пользователя и выполненное технико-экономическое обоснование разработки и использования автоматизированной системы.

В ходе дипломного проектирования были выделены, спроектированы и разработаны основные функции системы, добавлены некоторые возможности, которые подразумевают дальнейшее развитие проекта.

Это можно не проверять, оно в отдельном доке

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет

ИТиУ

Кафедра

ИТАС

Специальность

1-53 01 02

Специализация

06

УТВЕРЖДАЮ

Зав. кафедрой

« 2 » апреля 2020 г.

### ЗАДАНИЕ

по дипломному проекту (работе) студента

Колесникова Владислава Гавриловича

(фамилия, имя, отчество)

1 Тема проекта (работы): Автоматизированная система управления полочным пространством супермаркета

утверждена приказом по университету от « 01 » апреля 2020 г. № 750-с

2 Срок сдачи студентом законченной работы 01 июня 2020

3 Исходные данные к проекту: Microsoft Windows не ниже 7 версии;  
СУБД – *mongodb*;

язык программирования – *Python*; Библиотека – ; Фреймворк – *Django*;

библиотека - *Express*;

Назначение разработки:

разработать автоматизированную систему управления полочным пространством супермаркета

4 Содержание пояснительной записки (перечень подлежащих разработке вопросов):

Введение

1 Анализ объекта автоматизации

2 Проектирование структуры автоматизированной системы управления полочным пространством супермаркета

3 Программная реализация автоматизированной системы управления полочным пространством супермаркета

4 Техничко-экономическое обоснование разработки и использования автоматизированной системы управления полочным пространством супермаркета

Это можно не проверять, оно в отдельном доке

5 Перечень графического материала (с точным указанием обязательных чертежей):

6 Содержание задания по технико-экономическому обоснованию.  
Технико-экономическое обоснование разработки и использования автоматизированной системы управления полочным пространством супермаркета

Задание выдал

Т. Л. Слюсарь

### КАЛЕНДАРНЫЙ ПЛАН

| Наименование этапов дипломного проекта<br>(работы)   | Объём<br>этапа, % | Срок<br>выполнения<br>этапа | Примечание |
|--|-------------------|-----------------------------|------------|
| Сбор и изучение материалов по теме дипломного проектирования. Написание раздела 1 пояснительной записки, расчет технико-экономического обоснования | 40                | 20.04.2020                  |            |
| Проектирование программного комплекса. Написание раздела 2 пояснительной записки и соответствующего графического материала                         | 60                | 04.05.2020                  |            |
| Реализация программного комплекса. Написание раздела 3 пояснительной записки и соответствующего графического материала                             | 80                | 18.05.2020                  |            |
| Оформление пояснительной записки и подготовка презентации  | 100               | 01.06.2020                  |            |

Дата выдачи задания

02.04.2020

Руководитель

В.Л. Арановский

Задание принял к исполнению

В.Г. Колесников

## СОДЕРЖАНИЕ

|   |    |
|---|----|
| Перечень условных обозначений, символов и терминов .....  | 5  |
| Введение .....  | 7  |
| 1 Анализ объекта автоматизации .....  | 9  |
| 1.1 Описание объекта автоматизации .....  | 9  |
| 1.2 Анализ современных методов автоматизации работы супермаркетов ..  | 14 |
| 1.3 Анализ существующих систем и инструментов .....   | 19 |
| 1.4 Постановка задачи .....   | 23 |
| 2 Проектирование структуры автоматизированной системы .....   | 24 |
| 2.1 Структура разрабатываемой автоматизированной системы .....  | 24 |
| 2.2 Проектирование автоматизированной системы с использованием<br>инструментальных средств .....  | 25 |
| 2.3 Информационное обеспечение автоматизированной системы .....   | 31 |
| 2.4 Алгоритмическое обеспечение .....   | 38 |
| 2.5 Техническое и системное программное обеспечение .....   | 40 |
| 2.6 Эргономическое обеспечение .....  | 40 |
| 2.7 Организационное обеспечение .....   | 41 |
| 3 Программная реализация автоматизированной системы .....   | 43 |
| 3.1 Обоснование выбора средств разработки .....   | 43 |
| 3.2 Структура программных средств автоматизированной системы .....  | 47 |
| 3.3 Разработка программного кода автоматизированной системы .....   | 48 |
| 3.4 Описание программной реализации отдельных частей<br>автоматизированной системы .....  | 50 |
| 3.5 Руководство пользователя .....  | 56 |
| 4 Техничко-экономическое обоснование разработки и использования<br>автоматизированной системы управления полочным пространством<br>супермаркета ..... | 66 |
| 4.1 Характеристика автоматизированной системы управления полочным<br>пространством супермаркета .....   | 66 |
| 4.2 Расчет инвестиций в разработку автоматизированной системы .....   | 67 |
| 4.3 Расчет результата разработки и использования автоматизированной<br>системы .....  | 69 |
| 4.4 Расчет показателей экономической эффективности разработки и<br>использования автоматизированной системы .....                                     | 73 |
| 4.5 Выводы об экономической эффективности и целесообразности<br>инвестиций .....  | 77 |
| Заключение .....  | 78 |
| Список использованных источников .....  | 79 |
| Приложение А (обязательное) Программный код модуля анализа данных ...   | 80 |
| Ведомость дипломного проекта .....  | 85 |

## ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ, СИМВОЛОВ И ТЕРМИНОВ

Торговый объект – имущественный комплекс, а также иное имущество, используемое для осуществления торговли, принадлежащие на праве собственности, праве хозяйственного ведения, праве оперативного управления или на ином законном основании торговым организациям и (или) индивидуальным предпринимателям.

Магазин – специально оборудованное стационарное здание или его часть, предназначенное для продажи товаров и оказания услуг покупателям и обеспеченное торговыми, подсобными, административно-бытовыми помещениями, а также помещениями для приема, хранения и подготовки товаров к продаже.

Киоск – оснащенное торговым оборудованием строение, не имеющее торгового зала и помещений для хранения товаров, рассчитанное на одно рабочее место продавца, на площади которого хранится товарный запас.

Универсальный магазин – предприятие розничной торговли, реализующее универсальный ассортимент продовольственных и непродовольственных товаров.

Супермаркет – магазин с торговой площадью от 650 до 4000 квадратных метров, в котором реализуется универсальный ассортимент продовольственных товаров и ограниченный ассортимент непродовольственных товаров методами самообслуживания, традиционного обслуживания, продажи товаров по предварительным заказам.

Гипермаркет – предприятие торговли, реализующее продовольственные и непродовольственные товары универсального ассортимента преимущественно по форме самообслуживания.

Планограмма – документ, в котором детально изображается выкладка товаров с точным указанием мест размещения на торговом оборудовании торгового предприятия ассортиментных позиций.

Мерчендайзинг – оптимизация системы торговли, связанная с подготовкой товаров, их рекламой, а также стимулированием торговой деятельности.

*UML (Unified Modeling Language)* – унифицированный язык объектно-ориентированного моделирования.

*IDEF0* – методология, которая представляет собой совокупность методов, правил и процедур, предназначенных для построения функциональной модели объекта какой-либо предметной области.

*IDEF3* – методология моделирования, использующая графическое описание информационных потоков, взаимоотношений между процессами обработки информации и объектов, являющихся частью этих процессов.

СУБД – совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

## ВВЕДЕНИЕ

В связи с постоянным развитием и ростом сферы торговли, а также с постоянным улучшением процессов автоматизации, рано или поздно эти компоненты должны связаться между собой, образовав композицию из сложных систем, взаимодействующих между собой для минимизации расходов на шаблонные операции и максимизации прибыли торгового предприятия.

Примером шаблонной операции может являться работа с полочным пространством супермаркета. Данный процесс включает в себя не только расстановку товаров на полках, но и контроль за их количеством, а также их правильной расстановкой. Если в небольшом торговом объекте с этим не возникает больших проблем, то с увеличением размеров объекта появляются сложности, которые требуют увеличения торгового персонала, что вызывает рост расходов. Довольно крупные предприятия изначально планируют систему, которая позволит минимизировать влияние этих процессов на работу торгового объекта. Однако для торговых объектов среднего размера иногда данный фактор не учитывается в полной мере.

Внедрение в торговый процесс разработок, направленных на облегчение выполнения повседневных и многоразовых операций, позволяет многократно сократить затраты на те или иные действия при разумных вложениях.

Одним из вариантов частичной автоматизации работы супермаркета может являться автоматизированная система управления полочным пространством. Прежде всего, данная система позволяет следить за общим состоянием магазина в целом и каждой полки по отдельности в режиме реального времени на основе производимых операций купли-продажи и выкладки товаров на полки. Кроме того, данная система позволяет собирать и анализировать данные, поступающие в процессе работы супермаркета, на основе чего, вырабатывается дальнейшая тактика работы с выдачей определенных рекомендаций.

Для разработки данной системы необходимо решить следующие задачи:

- исследовать преимущества и недостатки существующих автоматизированных систем такого же или похожего типа;
- определить требования к автоматизированной системе;
- спроектировать автоматизированную систему;
- экономически либо социально обосновать разработку и реализовать автоматизированную систему.



Преимуществом данной автоматизированной системы является база данных, которая будет заполняться данными о товарах, их количестве на полках и складе, а также точным или приблизительным сроком годности. Кроме того, система обладает возможностью анализировать ежедневные операции с товарами, на основе чего строится статистика работы супермаркета, и настраиваются показатели на очередной день его работы.

Тема дипломного проекта: «Автоматизированная система управления полочным пространством супермаркета». Целью данного дипломного проекта является разработка и реализация автоматизированной системы управления полочным пространством супермаркета. Система позволит следить за общим состоянием магазина и каждой полки по отдельности, а также выдавать статистические данные о работе магазина с последующей настройкой параметров, что позволит сократить расходы на персонал.

Дипломный проект выполнен самостоятельно, проверен в системе «Антиплагиат». Процент оригинальности соответствует норме, установленной кафедрой (68%). Цитирования обозначены ссылками на публикации, которые указаны в разделе «Список использованных источников».

# **1 АНАЛИЗ ОБЪЕКТА АВТОМАТИЗАЦИИ**

## **1.1 Описание объекта автоматизации**

Принято различать торговые объекты по их видам и типам, которые, в свою очередь, зависят от различных факторов. Существует большое количество разновидностей магазинов. В основе их классификации лежат следующие признаки:

- тип ассортимента;
- уровень цен;
- форма обслуживания потребителей;
- местонахождение.

Каждый из них оказывает существенное влияние на прибыль бизнеса, но, в зависимости, например, от региона, экономической ситуации в данный период времени и прочих субъективных факторов, содержит те или иные признаки [1].

Большое влияние на тип торгового объекта оказывает его местоположение. Например, в таких местах, как остановки общественного транспорта, наиболее распространены киоски. Бóльшим разнообразием отличаются места, где чаще всего можно заметить массовое скопление людей. В таких местах обычно располагаются торговые центры, которые могут иметь как гигантские универсальные магазины, так и маленькие продуктовые магазины. Также существует такой класс гипермаркетов, которые могут располагаться за пределами города или на его границе. Обычно, такая форма торгового объекта предлагает широкий ассортимент товаров разнообразных классов и видов. Предназначены такие гипермаркеты преимущественно для людей, которые посещают магазин раз в несколько недель, а иногда и реже.

На данный момент размер торгового объекта и его ассортимент определяется, прежде всего, наличием спроса на тот или иной товар. Так, если спрос на строительные материалы преобладает в определенной местности, вероятнее всего, там может появиться специализированный магазин, который будет удовлетворять нужды покупателей.

Исходя из спроса, чаще всего происходит регулирование цен на ту или иную продукцию. Так, например, при достаточном количестве товара в разных торговых объектах и одинаковом спросе на него, при условии, что эти торговые объекты находятся неподалеку друг от друга, выиграет тот магазин, у которого цена на товар будет более привлекательной, нежели в другом. К сожалению, несмотря на регулирование цен, существуют ситуации, в которых

цена на товар может варьироваться в зависимости от предложения данного товара. Так, например, при дефиците товара определенного вида его цена будет большей, чем он мог бы стоить на самом деле. Чаще всего такие ситуации происходят в малых городах и деревнях, где товары сами по себе являются дефицитными в таких местах.

При достаточном количестве торговых объектов в больших городах иногда выигрывает тот, который является более дружелюбным и подходящим для клиента. То есть если магазины предлагают определенный товар по примерно одинаковой цене, может выиграть тот, в котором, по мнению покупателя, персонал более снисходителен и внимателен к покупателю, даже если цена в этом торговом объекте немного больше, чем в другом. Кроме того, если в магазине товары расположены таким образом, что найти нужную позицию бывает довольно сложно, а иногда и невозможно, либо торговое пространство распределено крайне неграмотно, такой магазин может потерять клиента в связи с данными обстоятельствами.

Учитывая выделенные факторы, необходимо проанализировать каждый вид торгового объекта, чтобы выделить тот вид, которому действительно будет необходима проектируемая автоматизированная система.

### **1.1.1 Розничный магазин**

В данном случае розничный магазин рассматривается как торговый объект, имеющий достаточно пространства для нахождения в нем от одного до нескольких продавцов и покупателей.

Розничные магазины имеют свои особенности по сравнению с универсальными магазинами: в небольших магазинах покупатель сразу же видит весь ассортимент товаров. Внутренний вид продовольственного магазина представлен на рисунке 1.1.

Также, кроме своих размеров, розничный магазин может специализироваться на предоставляемых услугах. Например, существуют специализированные магазины, предоставляющие товары определенного вида или класса, такие как магазины стройматериалов, ювелирные магазины, магазины бытовой техники, магазины косметики и другие. Внутренний вид магазина стройматериалов представлен на рисунке 1.2.

Так как в магазине могут присутствовать лишь несколько продавцов и покупателей, а ассортимент товаров зачастую представляет собой небольшую группу, относительно универсальных магазинов, данный вид торговых

объектов является довольно простым во взаимодействии между покупателями и продавцами.



Рисунок 1.1 – Продовольственный магазин



Рисунок 1.2 – Магазин строительных материалов

Проанализировав выделенные преимущества и недостатки данного вида торговых объектов, можно прийти к выводу, что небольшие магазины могут нуждаться только в автоматизации составления планogramм. Глубокий и расширенный анализ продаж и оптимизации расположения и количества товаров на полках магазина не требуются, так как расстановкой товаров на полках и их правильным расположением занимается продавец в течение рабочего дня.

### 1.1.2 Супермаркет

Супермаркет является одним из видов универсальных магазинов. В данном виде торговых объектов присутствует большой ассортимент продовольственных товаров и относительно небольшой ассортимент непродовольственных товаров, что уже дает ему преимущество перед розничными магазинами и киосками. Внутренний вид супермаркета представлен на рисунке 1.3.



Рисунок 1.3 – Супермаркет

Для работы целого супермаркета чаще всего нанимается персонал, представляющий собой группу специально обученных людей, состоящую из начального, производственного, среднего уровней и топ-менеджмента [2].

В супермаркете присутствуют большие стеллажи и длинные полки с продуктами. Нехватка продуктов на полках может являться как следствием недостаточного количества товаров на складе, так и следствием человеческого фактора, в результате которого продукты не расставлены на полках из-за невнимательности или нехватки времени.



В период акций и предложений, контроль за продукцией должен повышаться, особенно – за акционной продукцией. Кроме того, работа супермаркета в разное время дня, как и в разное время года, будет отличаться. Это тоже стоит учитывать при планировании нагрузки персонала.

Выделив определенные достоинства и недостатки данного вида торговых объектов, можно сделать вывод, что супермаркетам может понадобиться автоматизированная система управления полочным пространством, которая будет анализировать поступающие данные о работе супермаркета, на основе чего можно будет оптимизировать его работу.

### 1.1.3 Гипермаркет

Гипермаркет, как и супермаркет, является универсальным магазином. Гипермаркеты намного больше супермаркетов. Из-за своих больших размеров, гипермаркет может сам по себе являться складом. Кроме больших размеров, гипермаркеты выделяются тем, что могут предоставить богатый ассортимент товаров разного вида, как продовольственных, так и непродовольственных. Внешний вид гипермаркета изображен на рисунке 1.4.



Рисунок 1.4 – Гипермаркет

Данный вид торговых объектов весьма сложен в управлении, поэтому гипермаркеты набирают в менеджеры людей с опытом работы в данной сфере. Гипермаркеты являются очень большим комплексом, требующим

тщательного и грамотного контроля за всеми его частями. Специфика гипермаркета подразумевает высокую степень механизации и автоматизации всех работ, связанных как со внутренней системой магазина, так и с логистикой [3].

Выделенный набор особенностей гипермаркета показывает, что данная автоматизированная система будет недостаточной для управления данным видом торгового объекта. Для контроля гипермаркета необходимо разрабатывать целый комплекс взаимодействующих между собой систем, что выходит за рамки разрабатываемого дипломного проекта.

## **1.2 Анализ современных методов автоматизации работы супермаркетов**

Современные методы автоматизации предоставляют возможность автоматизировать многие процессы работы магазинов. Системы и средства позволяют внедрить свои компоненты в уже существующие информационные технологии, которые используются в супермаркетах. Некоторые решения требуют только программного обеспечения, но есть и такие системы, для работы которых необходимо еще и аппаратное обеспечение. Кроме того, популярность набирают облачные решения, предоставляющие функционал даже без дополнительных закупок или перенастроек оборудования.

### **1.2.1 Программные решения**

Многие компании изначально предоставляли большое разнообразие программных продуктов, поставляемых прямым на компьютер заказчика. Данные решения позволяли выполнять именно заказанные задачи прямым «из коробки», то есть они предоставляли тот функционал, который был нужен заказчику, без дополнительных установок другого программного обеспечения.

Такие программные продукты пользуются популярностью по сей день благодаря простоте их приобретения и использования, хотя в них и есть явные недостатки, которые могут повлиять на дальнейшее развитие предприятия. Так, например, при заказе программного продукта, могут сложиться трудности при его установке и сопровождении, так как иногда полученный продукт может конфликтовать с другими предустановленными программными продуктами. Кроме того, проблемы могут возникнуть со

стороны заказчика при его нежелании обновлять продукт по тем или иным причинам, оставаясь при этом на старой версии.

Одним из таких решений является система *S-Market*. Она представляет собой систему управления торговым предприятием. Данное решение предоставляет полнофункциональную систему управления товародвижением, а также автоматизацию учета товаров в магазине, контроля прихода товаров и их остатков, анализа эффективности продаж. Стартовая страница сайта системы *S-Market* представлена на рисунке 1.5.



Рисунок 1.5 – Стартовая страница сайта системы *S-Market*

Система состоит из набора функциональных модулей. Наиболее интересным является модуль складского учета. Он предоставляет автоматизацию в виде информации о всех поступающих и уходящих на полки товарах, загружаемых в базу данных администратором системы. Кроме программного обеспечения *S-Market* предоставляет поддержку аппаратного комплекса, автоматизирующего учет продаваемых товаров и некоторых операций с ними.

*S-Market* также включает в себя аналитический модуль, который предоставляет возможность расчета скорости продаж, просмотра товародвижения по карточкам товаров, по клиентам и по поставщикам, анализ продаж и расчет себестоимости товаров и ежедневных остатков.

### 1.2.2 Облачные решения

В отличие от программных решений, облачные решения предоставляют больше возможностей как для заказчика, так и для разработчика данного продукта. Существенными достоинствами данного решения являются такие факторы, как неограниченный объем дискового пространства, оперативной памяти и количества процессоров, доступ к продукту через веб-браузер, отсутствие зависимости от дорогого дополнительного оборудования, отсутствие нужды на развертывание инфраструктуры. Также, компании-заказчику не требуется выделять дополнительные средства на обучение



персонала использованию продукта, а сами облачные системы обслуживаются в основном высококвалифицированными специалистами, что оказывает весьма положительное влияние на качество предоставляемых услуг.

Разумеется, облачные решения не лишены изъянов. Наиболее остро стоят вопросы фильтрации данных и безопасности их передачи. Заказчик не всегда готов доверить стороннему провайдеру все свои данные, из-за чего приходится тщательно отбирать, какие данные можно передавать и хранить на сервере. Также, от заказчика будет требоваться постоянный доступ в интернет. Еще одним недостатком является возможность потерять данные в результате произошедшего сбоя у поставщика услуг.

Примером облачного решения является система автоматизации розничной торговли *CloudShop*. В качестве особенностей данной системы можно выделить ведение автоматизированного учета продукции. Все данные по поступлению и реализации товара поступают в базу данных, что в значительной степени упрощает все организационные процессы, необходимые для оперативной работы торговой точки. Страница системы *CloudShop* представлена на рисунке 1.6.

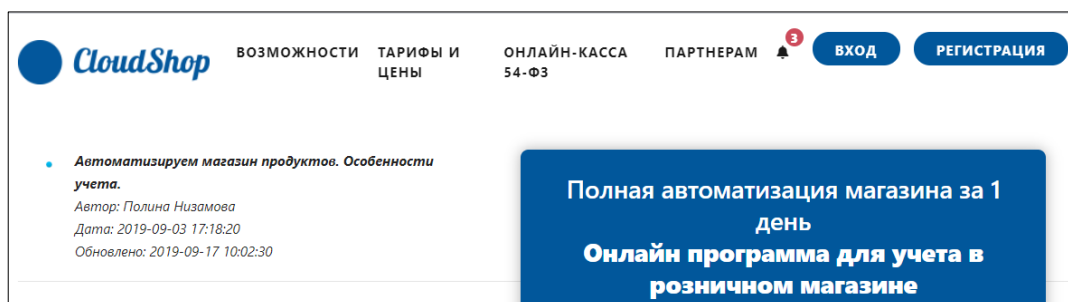


Рисунок 1.6 – Страница системы *CloudShop*

Алгоритм действий системы *CloudShop*:

- Для каждой товарной позиции создается карточка товара, которая содержит все данные по поступлению и реализации данного товара;
- Реализация продукта производится путем выбора карточки конкретного товара через терминал и отражения сведений об объеме покупки. Если позиций в корзине покупателя несколько, данные с нескольких карточек оформляются одним чеком;
- Как только продавец получил деньги от покупателя, покупка регистрируется, пробивается чек, и все данные по ней фиксируются в базе данных;

– Поскольку продукты имеют ограниченный срок годности, система оповещает о наличии товаров, срок реализации которых подходит к концу, или уже истек. В первом случае, продукты необходимо продать со скидкой, во втором – списать;

Также система позволяет вести контроль остатков продуктов на складе, а, следовательно, своевременно пополнять запасы и исключать позиции, на которые низок спрос [4].

### **1.2.3 Аппаратные решения**

Аппаратное решение практически всегда связано с программным обеспечением. Сама по себе аппаратура может нести мало смысла без какой-либо обработки или анализа данных. В этом плане, техническая составляющая супермаркета может работать в связке как с обычным программным решением, так и с облачным. Однако, для учета количества товаров и их правильного перемещения между супермаркетом и его складом, зачастую бывают необходимы аппаратные решения.

Преимуществ от внедрения аппаратных средств довольно много. Прежде всего, это позволяет автоматизировать многие процессы при работе с товарами и их количеством. Кроме того, техническая составляющая дает возможность легче и быстрее вносить и редактировать данные о том или ином товаре, а также данные о количестве проданных товаров, что способствует правильному подсчету аналитических данных.

К сожалению, во внедрении аппаратной составляющей также присутствуют и некоторые недостатки. Прежде всего, препятствием для внедрения техники в работу супермаркета может являться его дороговизна. Аппаратные решения стоят немалых денег, и если внедрять их не на этапе начала работы супермаркета, придется просчитать целесообразность производимых на это затрат. К недостаткам также можно отнести сложность работы с аппаратной составляющей, то есть для правильного использования аппаратуры персоналу будет необходимо пройти некоторое обучение. Также, аппаратное решение по своей сути предполагает наличие какого-либо программного обеспечения, поэтому супермаркету в любом случае придется внедрять дополнительно и программное решение, что также может предполагать некоторые затраты.

Типичными аппаратными решениями для ведения учета товаров супермаркета являются терминалы сбора данных и сканеры штрих-кодов.

Терминал сбора данных – это портативный компьютер, необходимый для оперативного сбора, обрабатывания и передачи данных о товарах и изделиях. Основное предназначение терминала сбора данных – сбор данных о товарах и грузах и их последующая передача в систему учета, уменьшение временных затрат на выполнение многих рутинных операций, а также снижение количества ошибок учета. Внешний вид терминала сбора данных представлен на рисунке 1.7.



Рисунок 1.7 – Терминал сбора данных

Одним из поставщиков данного решения является компания ШТРИХ-М СПб. Терминалы для сбора данных, представленные в компании, обладают широким набором функций и максимально просты в применении. Использование таких приборов дает возможность обеспечить дополнительные удобства при обслуживании покупателей и соответственно существенно увеличить объем продаж [5].

Сканер штрих кода — это устройство, которое обеспечивает сканирование штрих-кода, перевод его графических элементов в цифровую последовательность, декодирование данных, проверку качества считывания и передачу полученной информации в компьютер, кассовый терминал [6]. Таким образом, данное устройство позволяет обеспечить обработку информации о каждом считываемом товаре для его последующего учета.

### 1.3 Анализ существующих систем и инструментов

В рамках анализа существующих аналогов и прототипов разрабатываемого программного комплекса были изучены имеющиеся на рынке системы схожего предназначения. Многие из них специализируются такой функции, как построение оптимального плана выкладки товаров на полке для повышения продаж и увеличения прибыли. Кроме того, они предоставляют некоторый анализ уровня продаж. Однако, не удалось выявить систему, которая настраивала бы свои параметры в процессе работы и выдавала информацию по текущему состоянию продуктов и товаров на полках, а также рекомендации к действиям в определенный момент времени. Относительно данных факторов, можно сделать вывод, что разрабатываемая автоматизированная система предоставляет уникальные возможности в своей сфере.

#### 1.3.1 *ABM Shelf*

Данная система представляет собой облачное решение, разработанное компанией *ABM Cloud*. *ABM Shelf* предоставляет возможность составлять планограммы, моделировать залы и схемы выкладки в зависимости от разнообразных факторов. Кроме того, предоставляется возможность анализировать продажи магазина, в зависимости от чего выдаются статистические данные.

Система *ABM Shelf* предоставляет такие возможности, как конструктор оборудования, работу с базой данных магазинов, аналитику продаж торгового зала, автоматизацию управления выкладкой и управления торговым оборудованием, разработку планограммы выкладки товара, удобную визуальную аналитику по данным продаж и выкладки с возможностью экспорта в *Excel* [7]. Стартовая страница сайта системы *ABM Shelf* представлена на рисунке 1.8.

Система предоставляет много функционала по управлению полочным пространством. Данные о расположении товаров предоставляются в виде изображений торгового зала и каждого стеллажа по отдельности. Расположение товаров просчитывается автоматическими алгоритмами, благодаря чему выдаются оптимальные варианты расстановки товаров.

Также система предоставляет управление торговым пространством розничной сети с помощью анализа данных по мерчендайзингу товара,

который предполагает анализ таких параметров, как текущий остаток товара, выкладка товара и оптимальное количество товара.

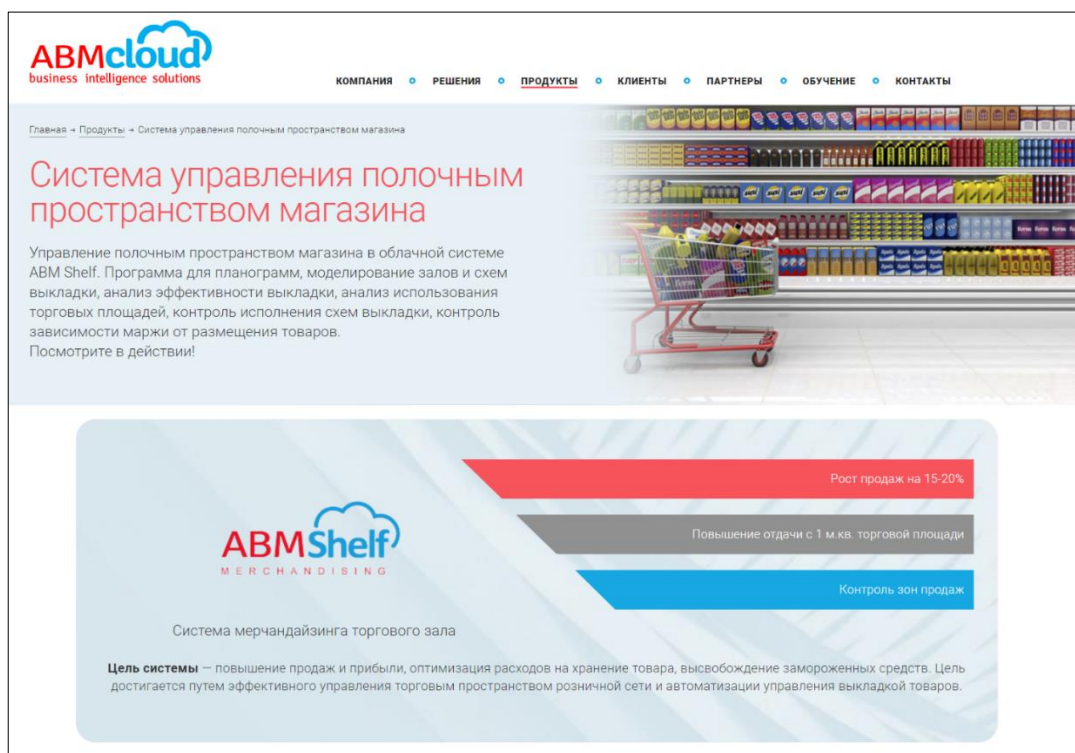


Рисунок 1.8 – Стартовая страница сайта системы *ABM Shelf*

К плюсам данной системы можно отнести:

- составление планограмм;
- моделирование торговых залов;
- аналитика продаж.

К минусам можно отнести:

- не всегда интуитивно понятный интерфейс;
- отсутствие взаимодействия со складом магазина.

### 1.3.2 *RS.ShelfSpace*

*Retail Suite Shelfspace* — это система управления полочным пространством магазина. Она позволяет автоматизировать и настроить эффективное управление торговым пространством розничной сети. Реализован сценарный подход, который позволяет одновременно управлять несколькими задачами, запущенными параллельно. Торговое пространство розничной сети должно быть эффективным и управляемым. Для того, чтобы

этого добиться, *RS.ShelfSpace* позволяет контролировать целый набор показателей визуального мерчандайзинга, отражающих качество использования торговых площадей [8].

Данная система обладает рядом преимуществ и недостатков. Прежде всего, она, возможно, могла бы предоставить больший функционал, однако по исходным данным видно, что система реализует лишь несколько общих вариантов автоматизации, без привнесения новизны в уже существующие методы. Так, например, данная система может составлять планограммы и возможные варианты расположения товаров только по заранее настроенным правилам, то есть это позволяет выставить определенный набор правил перед началом работы, относительно которого система выдаст возможные варианты без последующей возможности перенастройки системы в процессе работы. Кроме того, так как данная система анализирует данные только в конце рабочего дня, она не может сигнализировать об отсутствии определенного товара на полке, поэтому в данном случае остается человеческий фактор отслеживания заполненности полок.

В дополнение можно сказать, что данная система имеет довольно громоздкий интерфейс, предоставляющий информацию о стеллажах в виде списка типа «дерево». Данное решение может сказаться на интуитивном понимании интерфейса в случае разрастания списка. Пример конфигурации магазина представлен на рисунке 1.9. Стартовая страница сайта системы *RS.ShelfSpace* представлена на рисунке 1.10.

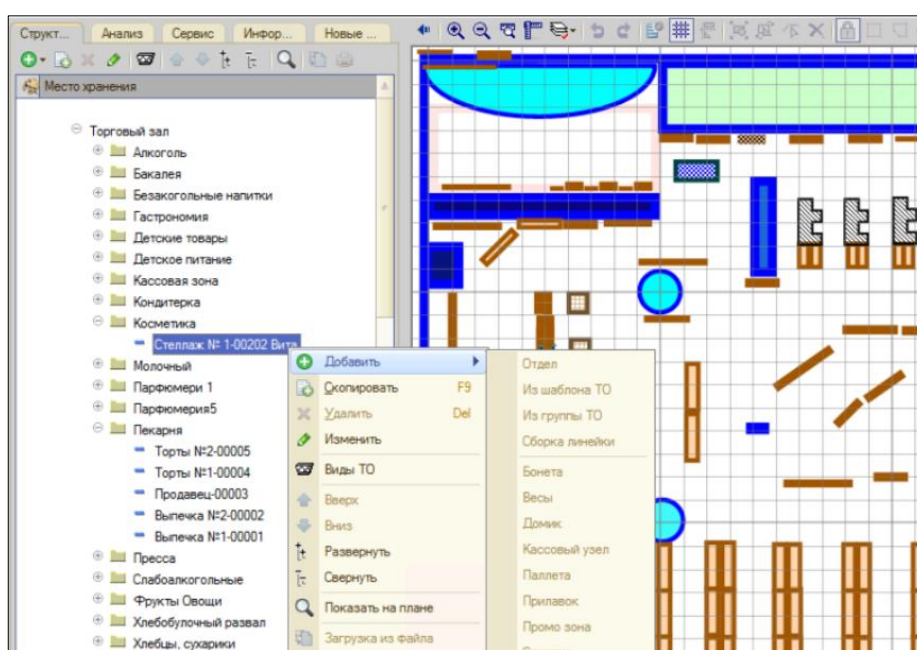


Рисунок 1.9 – Пример конфигурации магазина



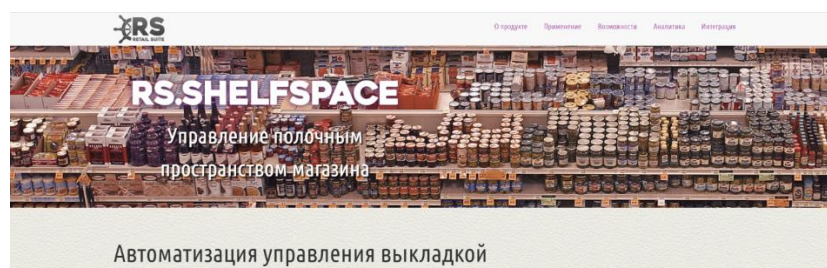


Рисунок 1.10 – Стартовая страница сайта системы *RS.ShelfSpace*

Преимущества системы *RS.ShelfSpace*:

- автоматизация мерчендайзинга;
- создание планограмм;
- анализ продаж.

Недостатки данной системы:

- анализ данных происходит только в конце рабочего дня;
- громоздкий интерфейс.

### 1.3.3 *PlanoManager*

Инструмент *PlanoManager* предназначен для создания планограмм и анализа их эффективности. Данное решение позволяет визуализировать торговое оборудование и размещение продукции, а также определить необходимый уровень товарного запаса для оценки и оптимизации выкладки. *PlanoManager* автоматически обновляет базу данных продуктов и автоматически строит планограммы на основе поступивших продуктов.

Данный инструмент поддерживает не так много функций, как предыдущие системы, однако он делает себя максимально эффективным в сфере планограмм. Кроме того, *PlanoManager* не предоставляет расширенной автоматизации на основе анализа продаж. Стартовая страница сайта инструмента *PlanoManager* представлена на рисунке 1.11.

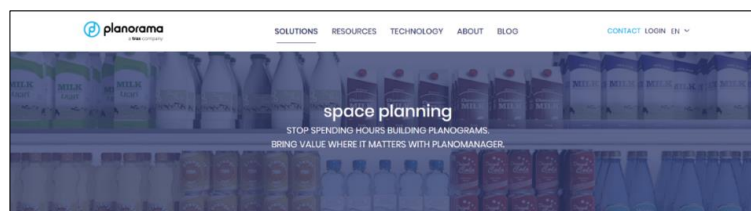


Рисунок 1.11 – Стартовая страница сайта инструмента *PlanoManager*

К преимуществам данного инструмента можно отнести:

- приятный, простой и интуитивно понятный интерфейс;
- расширенный функционал для построения планограмм;
- автоматическое обновление базы данных продуктов (на основе данных, предоставляемых пользователем);
- визуализация планограмм.

К недостаткам можно отнести:

- отсутствие расширенного анализа данных;
- отсутствие автоматизации контроля полочного пространства.

#### **1.4 Постановка задачи**

Разработать автоматизированную систему управления полочным пространством супермаркета, которая позволит автоматизировать ряд задач по работе с товарами, сократит расходы на персонал и улучшит качество обслуживания клиентов.

Система должна обладать возможностью визуализации текущего состояния супермаркета, анализировать поступающие данные и настраивать параметры своей работы в соответствии с полученными после обработки данными.

Система должна иметь базу данных для хранения и обработки поступающих данных о товарах и их перемещению по ключевым точкам супермаркета, а также для хранения и предоставления данных о статистике работы самой системы.

Необходимо, чтобы данная система была доступна для персонала супермаркета и обладала возможностью разграничения прав доступа к отдельным ее частям: управляющий персонал должны иметь доступ ко всем компонентам системы, авторизованные пользователи и рабочий персонал – к страницам «Состояние склада» и «Состояние магазина», неавторизованные пользователи – только к страницам «О системе» и «Авторизация». Доступ к системе должен предоставляться через веб-интерфейс.



## 2 ПРОЕКТИРОВАНИЕ СТРУКТУРЫ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ

### 2.1 Структура разрабатываемой автоматизированной системы

Разрабатываемая автоматизированная система должна быть доступна пользователям в сети интернет. Кроме того, системе необходимо наличие базы данных. Исходя из этих фактов, для реализации системы была выбрана модель взаимодействия клиент-сервер.

Концепция клиент-сервер подразумевает взаимодействие двух сторон: клиента и сервера. В качестве клиента может выступать заказчик той или иной услуги, а в качестве сервера – поставщик услуг. Клиентская часть решает задачи отображения контента, серверная же часть, в свою очередь, содержит бизнес логику, основные алгоритмы и берет на себя большую часть вычислительных нагрузок. Кроме того, серверная часть осуществляет связь с базой данных приложения. Типичным клиентом является браузер, в качестве сервера может выступать как удаленный *HTTP* сервер, так и локальный веб-сервер.

К преимуществам клиент-серверной архитектуры можно отнести:

- программный код клиентского приложения и серверного разделен;
- минимальные системные требования к компьютерам пользователей, т.к. вся аналитическая нагрузка вынесена на сервер.

Одним из недостатков данной архитектуры является то, что недоступность сервера по любой причине парализует работу всего программного комплекса.

Пользователь взаимодействует с клиентской частью приложения, которая при необходимости формирует запросы и обращается на сервер за данными, после чего принимает ответ и отображает данные.

Серверная часть, при поступлении на нее запроса, совершает обработку введенных данных, в большинстве случаев обращается в базу данных за данными, из которых, после анализа, формирует ответ и отправляет его на клиентскую часть.

Архитектура клиент-сервер не делит машины на только клиент или только сервер, а скорее позволяет распределить нагрузку и разделить функционал между клиентской частью и серверной. Архитектура приложения представлена на рисунке 2.1.

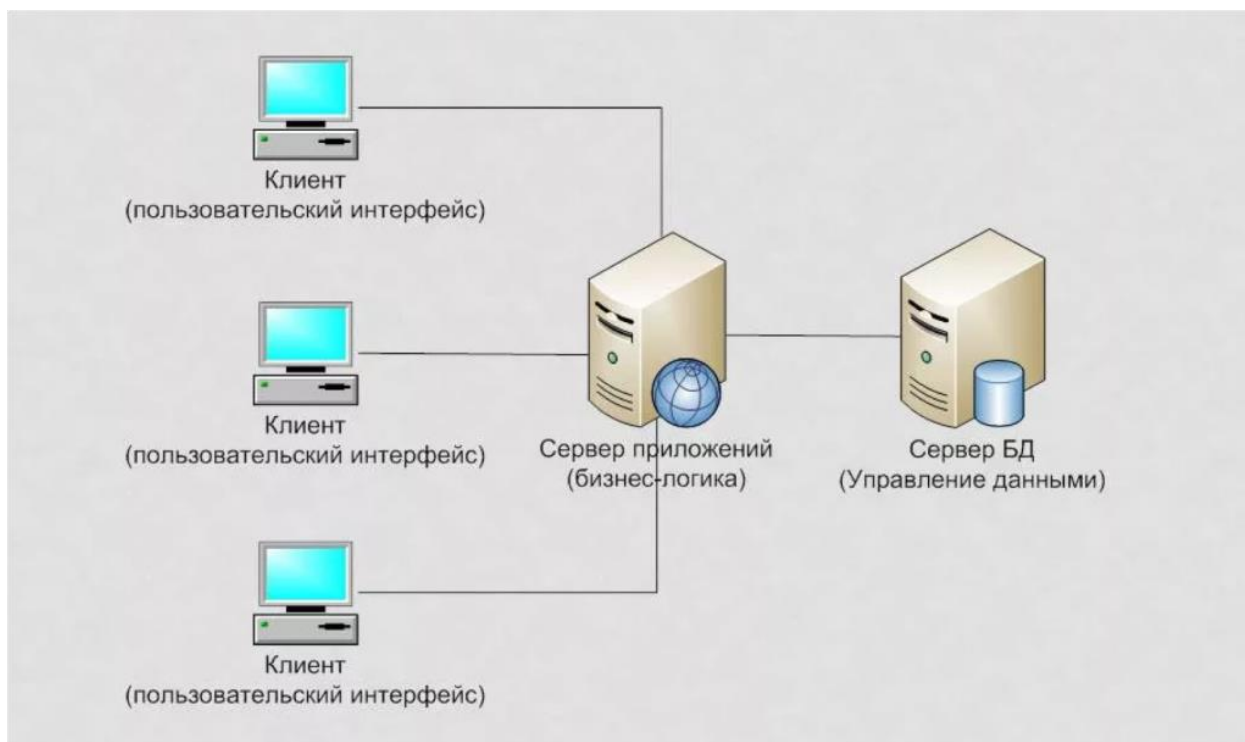


Рисунок 2.1 – Архитектура клиент-сервеного приложения

## 2.2 Проектирование автоматизированной системы с использованием инструментальных средств

### 2.2.1 Проектирование автоматизированной системы с использованием языка *UML*

*UML* (*Unified Modeling Language*) – это унифицированный язык объектно-ориентированного моделирования. Он служит для наглядного представления функциональности разрабатываемого программного продукта.

Для построения наиболее общей и абстрактной концептуальной модели системы используется диаграмма вариантов использования. Она описывает взаимодействие объектов автоматизированной системы между собой в процессе использования ее по назначению. Диаграмма вариантов использования автоматизированной системы представлена на рисунке 2.2.

В проектируемой автоматизированной системе можно выделить пятерых актеров: неавторизованный пользователь, персонал, менеджер, системный администратор, программное и аппаратное обеспечение.

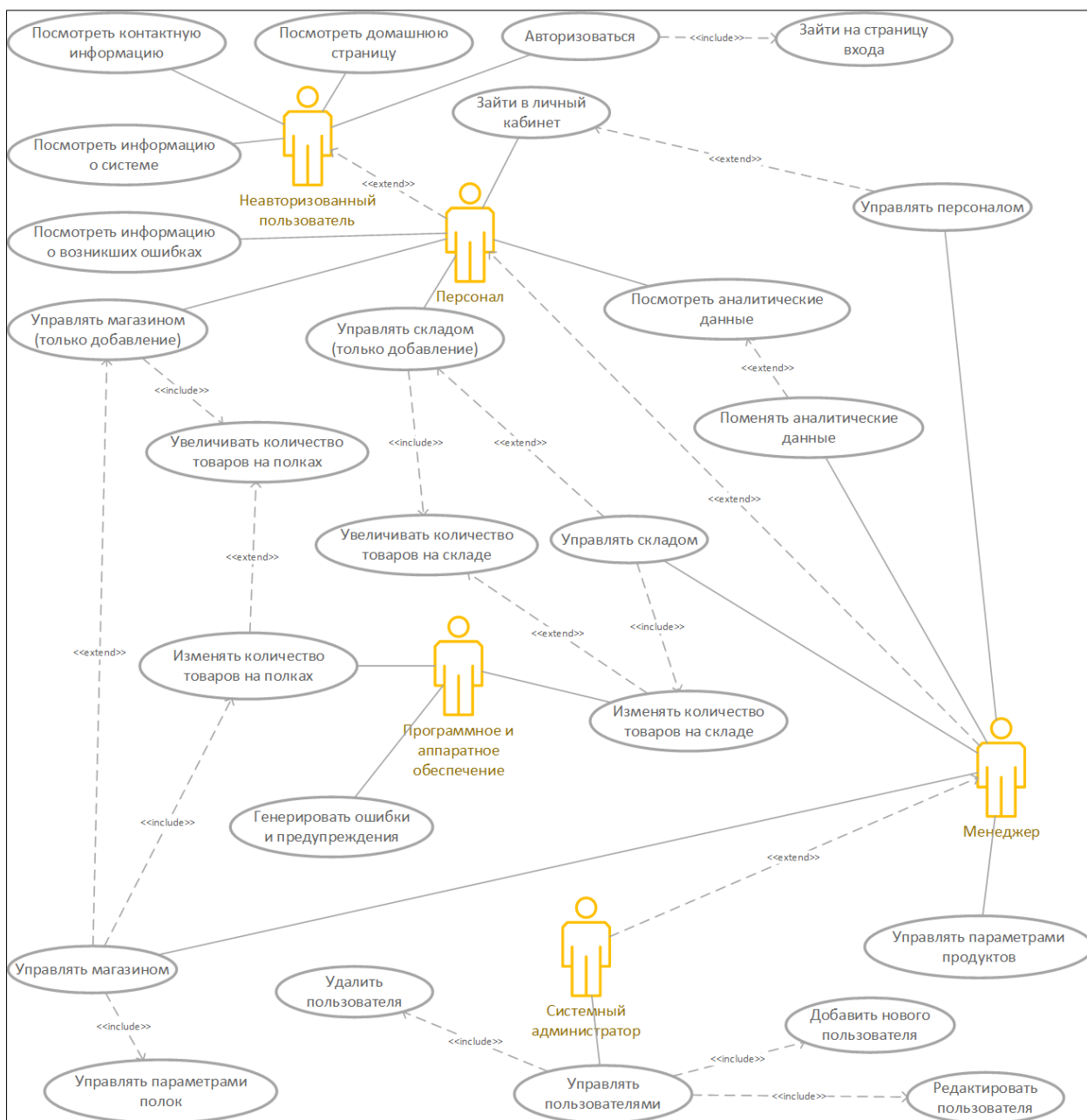


Рисунок 2.2 – Диаграмма вариантов использования автоматизированной системы

У каждого из актеров имеются свои варианты использования, которые также называются прецедентами. Прецедент – это типичное взаимодействие пользователя с системой, которое при этом:

- описывает видимую пользователем функцию;
- может представлять различные уровни детализации;
- обеспечивает достижение конкретной цели, важной для пользователя.

Рассмотрим варианты использования каждого актера. Возможности автоматизированной системы, доступные неавторизованному пользователю:

- авторизация, которая включает возможность перехода на страницу авторизации;
- просмотр домашней страницы;
- просмотр контактной информации владельца системы;
- просмотр общей информации о системе.

Персонал имеет все возможности неавторизованных пользователей. Также ему доступны такие функции, как:

- вход в личный кабинет для просмотра индивидуальной рабочей информации;
- просмотр аналитических данных системы;
- ограниченное управление складом, которое подразумевает увеличение количества товаров на складе;
- ограниченное управление магазином, которое подразумевает увеличение товаров на полках;
- просмотр информации о возникших ошибках и предупреждениях.

Менеджер расширяет функционал персонала с добавлением таких возможностей, как:

- управление персоналом в рамках системы;
- изменение аналитических данных и параметров системы;
- полное управление складом;
- полное управление магазином;
- управление продуктами, с которыми взаимодействует система.

В свою очередь, системный администратор обладает всеми перечисленными возможностями, включая управление пользователями системы. То есть именно системный администратор имеет возможность изменять параметры пользователей, а также добавлять и удалять их.

Кроме того, система подразумевает наличие сущности, не относящейся к живым аналогам, но обладающей возможностью субъективного взаимодействия с некоторыми частями системы и являющейся синтезом программного и аппаратного обеспечения. Данный актер обладает такими возможностями, как изменение количества товаров на полках и на складе, а также генерация ошибок и предупреждений, возникающих в процессе работы системы.

### 2.2.2 Функциональная структура автоматизированной системы

Для формализации и описания бизнес-процессов используется методология функционального моделирования *IDEF0*. В *IDEF0* рассматривается функциональная структура объекта, т.е. выполняемые объектами действия и логические отношения между ними, а не их временная последовательность.

Функциональная модель *IDEF0* представляет собой набор блоков, каждый из которых имеет входы и выходы, управление и механизмы. Наиболее важная функция расположена в верхнем левом углу. Соединяются функции между собой при помощи стрелок и описаний функциональных блоков.

Контекстная диаграмма разрабатываемой автоматизированной системы изображена на рисунке 2.3.

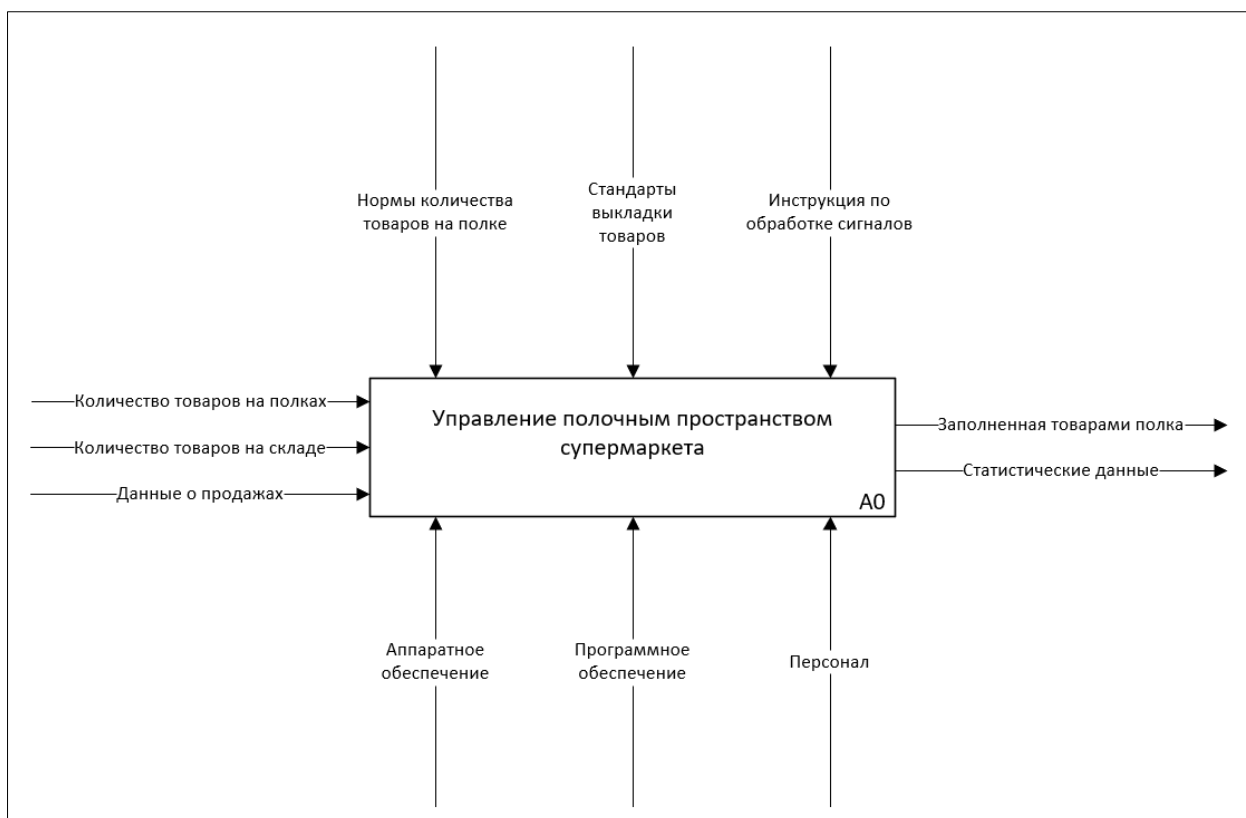


Рисунок 2.3 – Контекстная диаграмма автоматизированной системы управления полочным пространством супермаркета

На вход системы подаются данные о количестве товаров на полках и на складе, а также данные о продажах, которые были произведены в определенный промежуток времени.

Правилами системы являются нормы количества товаров на полке, стандарты выкладки товаров и инструкции по обработке сигналов. Первые два правила дают описание регулирования товаров в зависимости от его избытка, недостатка и расположения. Третье правило задает порядок действий при возникновении определенных сигналов в процессе работы системы.

Механизмами управления являются аппаратное обеспечение, программное обеспечение и персонал супермаркета. Аппаратное обеспечение позволяет производить учет товаров в супермаркете при продаже и поступлении. Программное обеспечение выполняет хранение данных о магазине, производит подсчет проданных и поступивших товаров, а также производит аналитические вычисления. Персонал выполняет контролирующую функцию, то есть следит за корректным выполнением задач, возложенных на систему, а также заполняет полки.

Результатом работы системы являются заполненные товарами полки и полученные статистические данные. Полки будут считаться заполненными в том случае, когда соблюдены все правила, заданные при работе системы, а также при отсутствии ошибочных сигналов. Статистические данные представляют собой общие итоговые результаты работы системы, используемые для дальнейшего анализа.

Чтобы более подробно описать процессы, происходящие внутри автоматизированной системы, используется диаграмма декомпозиции, представленная на рисунке 2.4.

На диаграмме изображены 9 работ, которые будут выполняться в процессе функционирования системы.

Перед началом работы программное обеспечение вычисляет оптимальное количество товаров на полках. Для этого необходимы данные о текущем количестве, а также о предыдущих продажах. Оптимальное количество вычисляется с ориентировкой на нормы допустимого количества товаров.

Вычисленные данные проходят на следующий этап, на котором аппаратное обеспечение во взаимодействии с программным подает сигнал либо об избыточном количестве товаров на полке, либо о недостаточном. Данные сигналы записываются в раздел данных, из которого будет формироваться статистический результат работы.

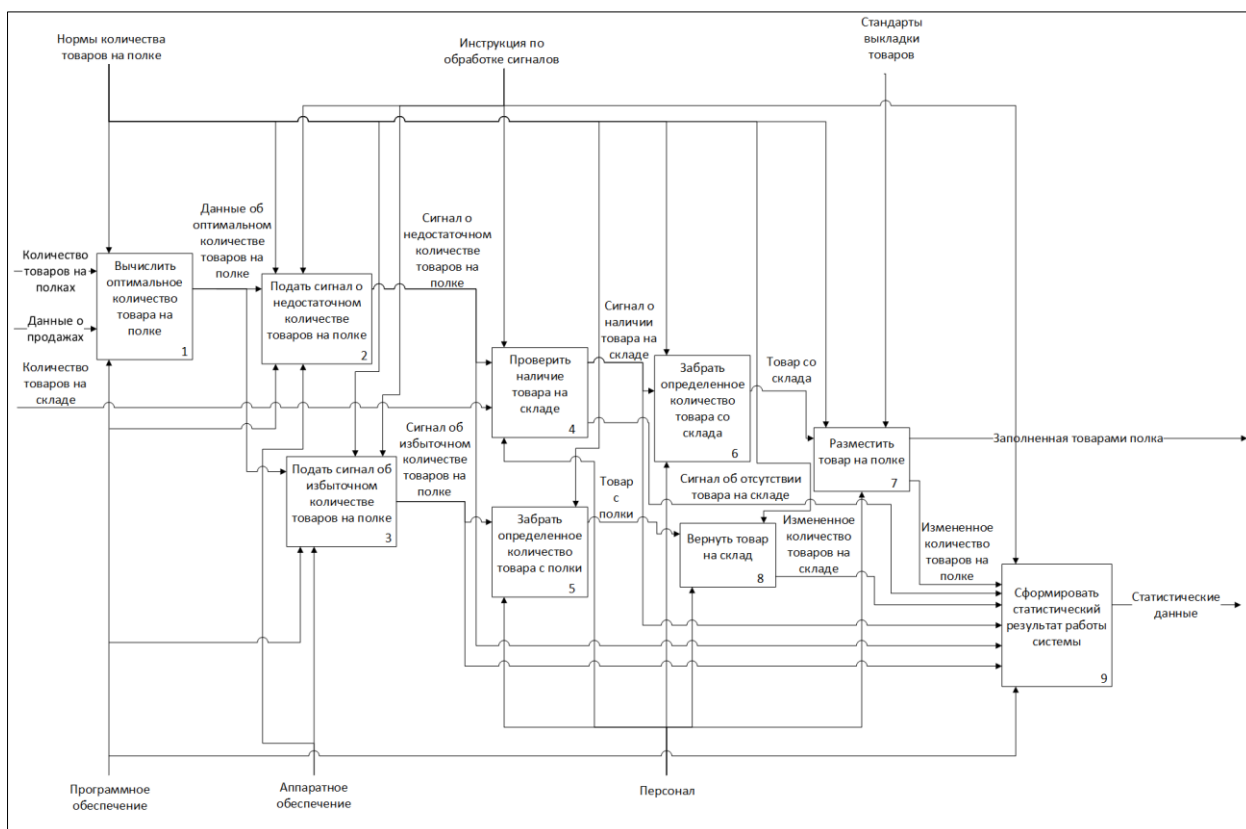


Рисунок 2.4 – Диаграмма декомпозиции автоматизированной системы управления полочным пространством супермаркета

При сигнале о недостаточном количестве товара на полке персоналом выполняется проверка наличия товара на складе. В случае отсутствия товара, подается сигнал, записывающийся в раздел статистики. При наличии товара также подается сигнал, после чего персонал забирает определенное количество со склада.

Полученный на складе товар размещается в соответствии со стандартами выкладки товаров на полке. Измененное количество товара записывается в окончательную статистику.

При сигнале об избыточном количестве товара на полке, персонал забирает товар с полки до тех пор, пока его количество не станет оптимальным. Товар с полки возвращается на склад. Измененное количество товара на складе передается в статистические данные.

По окончании выполнения данного процесса формируется статистический результат работы системы. На выходе получают сформированные на основе работы системы статистические данные.

Исходя из того, что в процессе работы автоматизированной системы подразумевается взаимодействие с информацией, имеет смысл построение

диаграммы на основе *DFD* методологии для этапа работы с базой данных. Такая диаграмма представлена на рисунке 2.5.

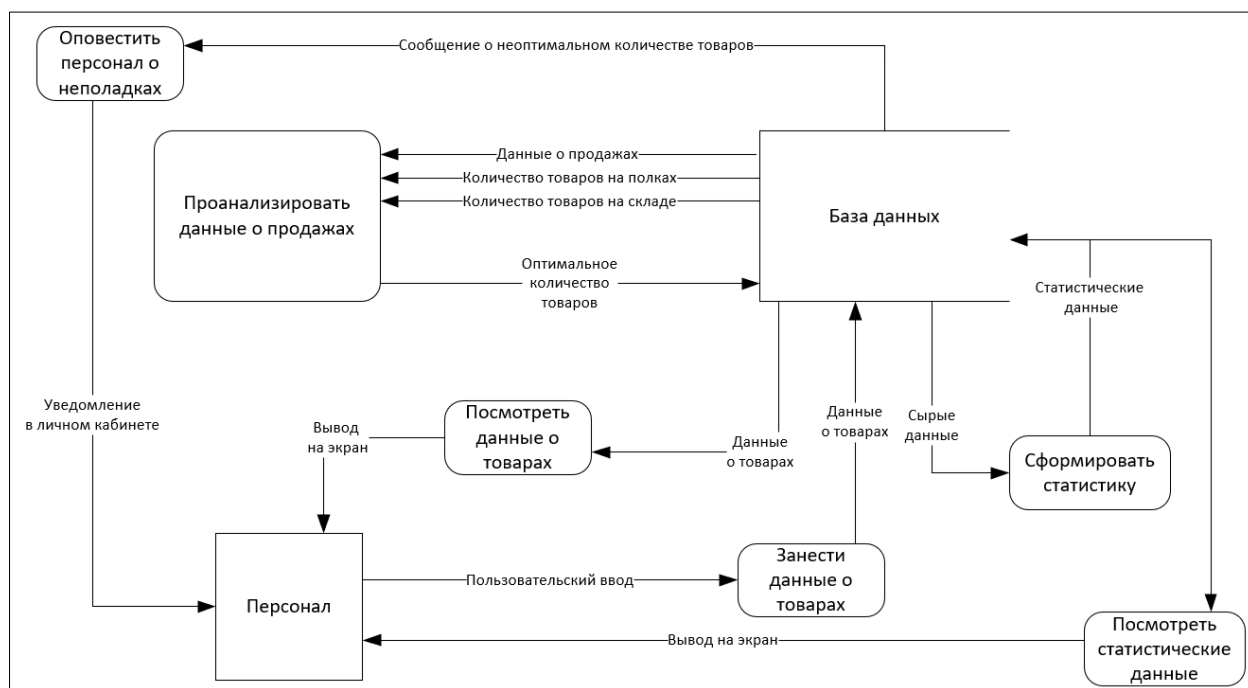


Рисунок 2.5 – Диаграмма декомпозиции процесса работы с базой данных

На диаграмме декомпозиции процесса работы с базой данных изображены потоки данных, поступающих на обработку, на ввод и в качестве результатов. Все данные поступают изначально от персонала. После этого различные модули достают необходимую информацию и обрабатывают ее в соответствии с определенными правилами. Результатирующими данными будут являться статистические данные, которые сохраняются в базу данных и выводятся на экран персоналу. Кроме того, в процессе работы системы персоналу будут приходить уведомления о тех или иных неполадках.

## 2.3 Информационное обеспечение автоматизированной системы

### 2.3.1 Входные данные

Разрабатываемая автоматизированная не предполагает наличие системы регистрации новых пользователей, так как количество пользователей ограничено персоналом супермаркета. Для создания нового пользователя используются возможности панели администратора, взаимодействовать с которой может системный администратор.



Входные данные для нового пользователя:

- логин пользователя;
- пароль пользователя;
- электронная почта;
- группа, к которой пользователь принадлежит;
- дата и время последнего входа.

Входные данные автоматизированной системы представляют собой некоторые качественные и количественные характеристики супермаркета и продуктов, необходимые для работы системы.

Входные данные по продуктам:

- код продукта;
- размеры продукта;
- название продукта;
- цена за единицу продажи;
- срок годности;
- вес единицы продажи.

Входные данные по параметрам полок:

- относящийся к полке продукт;
- грузоподъемность полки;
- размеры полочного пространства;
- количество единиц продажи;
- название полки;
- название секции, к которой относится полка.

Дополнительные входные данные относятся к ячейкам склада, такие как продукт, относящийся к ячейке, дата поставки, количество продукта в ячейке, и акционным товарам, такие как продукт, на который распространяется акция, полка, к которой относится данная акция, сроки проведения акции и цена единицы продукта по акции.

Источником входных данных являются персонал и менеджер супермаркета. Некоторые входные данные, такие как даты и необязательные поля, генерируются автоматически. Периодичность поступления данных отсутствует, так как данные могут быть занесены в любое время любым пользователем системы.

## 2.3.2 Проектирование базы данных

Разрабатываемая автоматизированная система взаимодействует с базой данных. База данных представляет собой совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняется в соответствии с правилами средств моделирования данных. В базе данных разрабатываемой автоматизированной системы содержится 8 сущностей: пользователи, продукт, магазин, акция, продажи, склад, сбои, статистика.

*Django* использует технологию объектно-реляционного отображения для упрощения взаимодействия базы данных и программного кода. Данная технология позволяет автоматизировать некоторые процессы создания сущностей базы данных и их экземпляров, представляя возможные поля сущностей в виде связанной модели. Такой подход упрощает создание связей между сущностями, а также позволяет упростить изменение полей и связей сущности.

Сущность «*User*» содержит информацию о каждом зарегистрированном пользователе. Данная сущность связана со стандартной сущностью «*User*», предоставляемой фреймворком *Django*, которая содержит такие поля, как имя пользователя, фамилия, логин, электронная почта, пароль, дата последнего входа в систему, а также связана с сущностью «*Groups*», которая предоставляет информацию о группе пользователя. Структура сущности «*User*» приведена в таблице 2.1.

Таблица 2.1 – Структура сущности «*User*»

| Поле         | Тип                       | Описание  |
|--------------|---------------------------|---|
| <i>Id</i>    | <i>AutoField</i>          | Идентификатор пользователя                        |
| <i>User</i>  | <i>OneToOneField (id)</i> | Идентификатор пользователя стандартной реализации |
| <i>Group</i> | <i>CharField</i>          | Группа пользователя в удобочитаемом виде          |

Сущность «*Failure*» содержит информацию о возникших в процессе работы системы ошибках, возможных причинах и способах устранения. Ошибки в таблице помечаются исправленными, если поле *is\_solved* содержит значение *True*. Кроме того, каждой ошибке может присваиваться пользователь, которому поручено ее устранить. За информацию о пользователе, которому выдано задание на устранение ошибки, отвечает поле *assignee*. Структура таблицы «*Failure*» приведена в таблице 2.2.

Сущность «*Product*» содержит информацию о продуктах, когда-либо заносимых в систему. Данная сущность предоставляет о параметрах продуктов, их текущей цене, сроке годности и других характеристиках. Ее структура приведена в таблице 2.3.

Таблица 2.2 – Структура сущности «*Failure*»

| Поле                     | Тип                    | Описание   |
|--------------------------|------------------------|--|
| <i>Id</i>                | <i>AutoField</i>       | Идентификатор неполадки  |
| <i>Assignee</i>          | <i>ForeignKey (id)</i> | Идентификатор пользователя, которому назначено решение неполадки |
| <i>Is_solved</i>         | <i>BooleanField</i>    | Статус решения ошибки  |
| <i>Possible_cause</i>    | <i>TextField</i>       | Описание возможного источника                                    |
| <i>Possible_solution</i> | <i>TextField</i>       | Описание возможного решения                                      |
| <i>Severity</i>          | <i>IntegerField</i>    | Серьезность ошибки   |
| <i>Text</i>              | <i>TextField</i>       | Описание ошибки  |
| <i>Timestamp</i>         | <i>DateTimeField</i>   | Дата и время возникновения ошибки                                |

Таблица 2.3 – Структура сущности «*Product*»

| Поле              | Тип                 | Описание                                 |
|-------------------|---------------------|--|
| <i>Id</i>         | <i>AutoField</i>    | Идентификатор продукта                   |
| <i>Digit_code</i> | <i>CharField</i>    | Цифровой код продукта                    |
| <i>Height</i>     | <i>FloatField</i>   | Высота продукта в миллиметрах            |
| <i>Length</i>     | <i>FloatField</i>   | Длина продукта в миллиметрах             |
| <i>Width</i>      | <i>FloatField</i>   | Ширина продукта в миллиметрах            |
| <i>Name</i>       | <i>CharField</i>    | Название продукта                        |
| <i>Price</i>      | <i>FloatField</i>   | Цена продукта                            |
| <i>Shelf_life</i> | <i>IntegerField</i> | Срок годности продукта                   |
| <i>Stackable</i>  | <i>BooleanField</i> | Возможность складывать продукт по высоте |
| <i>Weight</i>     | <i>FloatField</i>   | Вес продукта                             |

Сущность «*Storage*» содержит информацию о продуктах, хранящихся на складе. Данная сущность связана с таблицей «*Product*» и предоставляет информацию о дате и времени последней поставки продуктов на склад, а также текущем количестве на складе. Ее структура представлена в таблице 2.4.

Таблица 2.4 – Структура сущности «*Storage*»

| Поле                     | Тип                    | Описание                               |
|--------------------------|------------------------|--|
| <i>Id</i>                | <i>AutoField</i>       | Идентификатор ячейки склада            |
| <i>Product</i>           | <i>ForeignKey (id)</i> | Идентификатор продукта                 |
| <i>DeliveryTimestamp</i> | <i>DateTimeField</i>   | Дата и время последней поставки        |
| <i>Product_count</i>     | <i>IntegerField</i>    | Текущее количество продуктов на складе |

Сущность «*Store*» содержит данные о каждой полке магазина. Данная сущность связана с таблицами «*Product*», «*User*» и «*Statistics*» и предоставляет информацию о пользователе, которому поручено работать с данной полкой, названии полки, названии секции, к которой относится полка, параметрах полки, продукте, который хранится на полке в данный момент, его количестве и последней загрузке, а также о максимально допустимом весе, который выдерживает полка. Структура данной таблицы представлена на рисунке 2.5.

Таблица 2.5 – Структура сущности «*Store*»

| Поле                     | Тип                    | Описание                                      |
|--------------------------|------------------------|---|
| <i>Id</i>                | <i>AutoField</i>       | Идентификатор полки                           |
| <i>Product</i>           | <i>ForeignKey (id)</i> | Идентификатор продукта                        |
| <i>Carrying_capacity</i> | <i>FloatField</i>      | Максимально допустимый вес продуктов на полке |
| <i>Height</i>            | <i>FloatField</i>      | Высота полки в миллиметрах                    |
| <i>Length</i>            | <i>FloatField</i>      | Длина полки в миллиметрах                     |
| <i>Width</i>             | <i>FloatField</i>      | Ширина полки в миллиметрах                    |
| <i>Last_charge</i>       | <i>DateTimeField</i>   | Дата и время последней загрузки полки         |
| <i>Product_count</i>     | <i>IntegerField</i>    | Текущее количество продуктов на полке         |
| <i>Section_name</i>      | <i>CharField</i>       | Название секции, в которой расположена полка  |
| <i>Shelf_name</i>        | <i>CharField</i>       | Название полки                                |

Сущность «*Stock*» хранит данные об акциях, предусмотренных в супермаркете. Эта сущность связана с таблицами «*Statistics*» и «*Product*» и предоставляет информацию о продукте, об акционной цене, дате и времени начала и конца акции. Структура данной сущности представлена в таблице 2.6.

Таблица 2.6 – Структура сущности «*Stock*»

| Поле                   | Тип                    | Описание                                |
|------------------------|------------------------|---|
| <i>Id</i>              | <i>AutoField</i>       | Идентификатор акции                     |
| <i>Product</i>         | <i>ForeignKey (id)</i> | Идентификатор продукта                  |
| <i>Start_timestamp</i> | <i>DateTimeField</i>   | Дата и время начала проведения акции    |
| <i>End_timestamp</i>   | <i>DateTimeField</i>   | Дата и время конца проведения акции     |
| <i>Stock_price</i>     | <i>FloatField</i>      | Цена продукта во время проведения акции |

Сущность «*Statistics*» содержит аналитические данные на каждый день работы супермаркета. Она связана с таблицами «*Store*» и «*Stock*» и предоставляет информацию о количестве произошедших неполадок, полке, для которой строится статистика, количестве проданных продуктов, цене на единицу продажи в тот день, а также проводились ли в данный день акции для заданной полки. Структура данной сущности приведена в таблице 2.7.

Таблица 2.7 – Структура сущности «*Statistics*»

| Поле                  | Тип                    | Описание                                     |
|-----------------------|------------------------|--|
| <i>Id</i>             | <i>AutoField</i>       | Идентификатор записи                         |
| <i>Shelf</i>          | <i>ForeignKey (id)</i> | Идентификатор полки                          |
| <i>Stock</i>          | <i>ForeignKey (id)</i> | Идентификатор акции                          |
| <i>Day</i>            | <i>DateField</i>       | Дата, на которую составлена статистика       |
| <i>Failures_count</i> | <i>IntegerField</i>    | Количество произошедших неполадок            |
| <i>Price_that_day</i> | <i>FloatField</i>      | Цена на продукт в заданный день              |
| <i>Sold_count</i>     | <i>IntegerField</i>    | Количество проданных продуктов в данный день |

На рисунке 2.6 представлена структурная схема используемой базы данных.

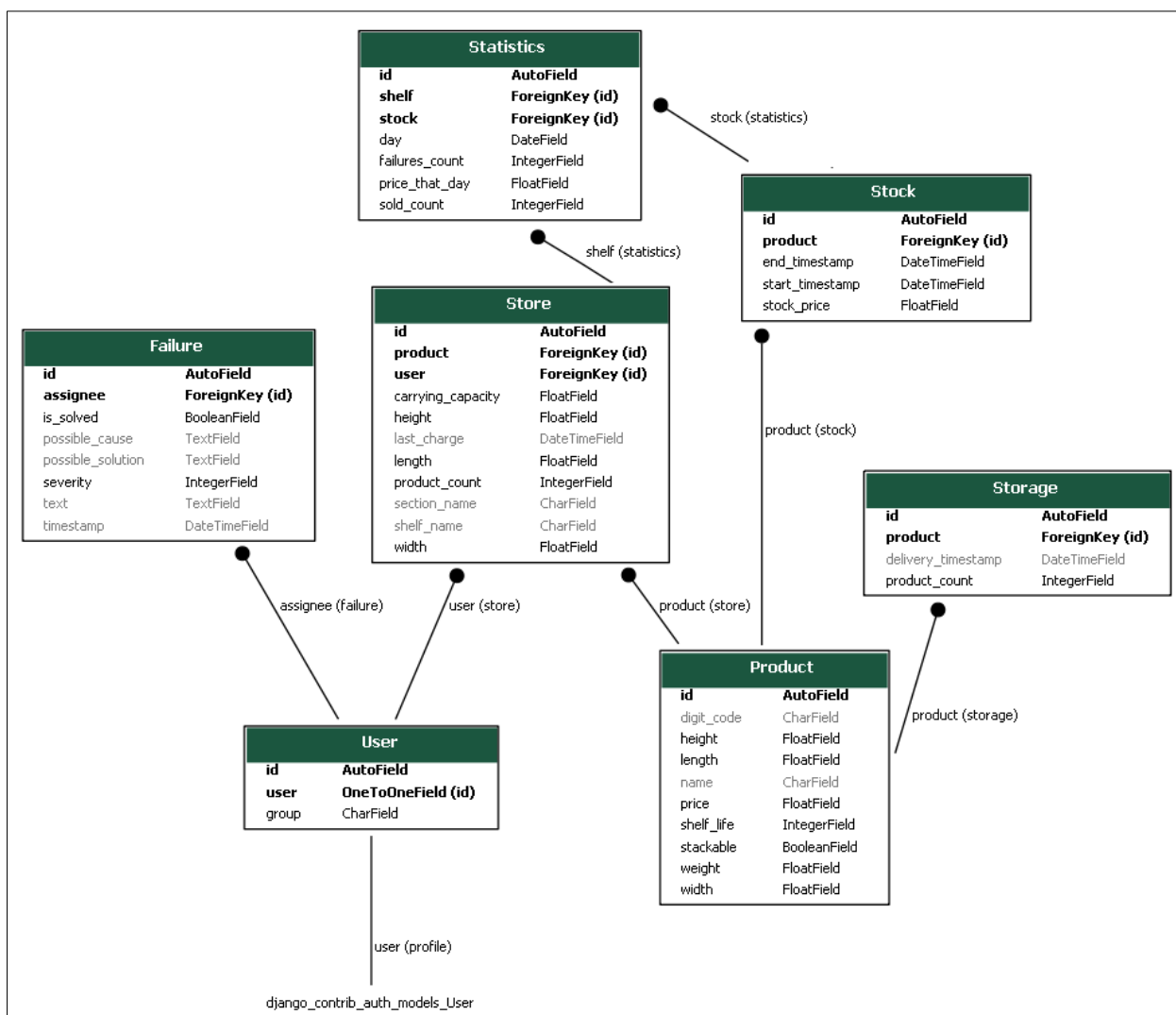


Рисунок 2.6 – Структурна схема базы данных

### 2.3.3 Выходные данные

Выходными данными автоматизированной системы являются информация о текущем состоянии супермаркета, оповещения о неполадках системы и статистические данные работы супермаркета в связке с системой.

В качестве информации о текущем состоянии супермаркета выступают:

- информация о заполненности полок;
- информация о заполненности склада;
- внесенная в систему информация о продуктах;
- информация о проводимых и проведенных акциях.

Оповещения о неполадках системы включают в себя:

- выдачу информации о неполадке персоналу;
- выдачу информации о разрешении проблем менеджеру.

Статистические данные состоят из:

- аналитических данных о работе супермаркета в каждый день;
- аналитических данных о работе супермаркета в определенный период.

## **2.4 Алгоритмическое обеспечение**

Система предоставляет поддержку наблюдения за товарами на полках. Она подразумевает участие человека в своем процессе. Исходя из этого, для работы с системой следует описать алгоритм взаимодействия.

Главный действующий персонаж в системе – персонал. Люди, которые зарегистрированы в системе, имеют общий алгоритм работы с ней. Прежде всего, перед началом работы им необходимо авторизоваться. После этого происходит регулярное наблюдение за количеством товаров на полках до конца рабочего дня.

При избыточном количестве товаров, пользователь, за которым закреплена полка в данный момент времени, производит отгрузку товара на склад с занесением данных об измененном количестве товаров на полке и на складе. При недостаточном количестве товаров на полке алгоритм становится слегка другим. Сначала пользователю необходимо проверить, есть ли данный товар на складе. При отсутствии товара, сотрудник оповещает об этом менеджера, после чего продолжает свой рабочий день, наблюдая за другими полками.

Кроме того, что пользователь сам следит за количеством товаров на полках, система также может выдавать оповещения о неполадках, на которые должен реагировать сотрудник. В оповещении система выдает рекомендуемое количество товаров на перенос на склад при избыточном количестве товаров, либо на дозагрузку полки при недостаточном количестве товаров. При выполнении рекомендаций системы и изменении количества товаров на полке с занесением данных в систему, неполадка считается устраненной и отмечается в базе данных как разрешенная.

Блок-схема функционирования автоматизированной системы представлена на рисунке 2.7.

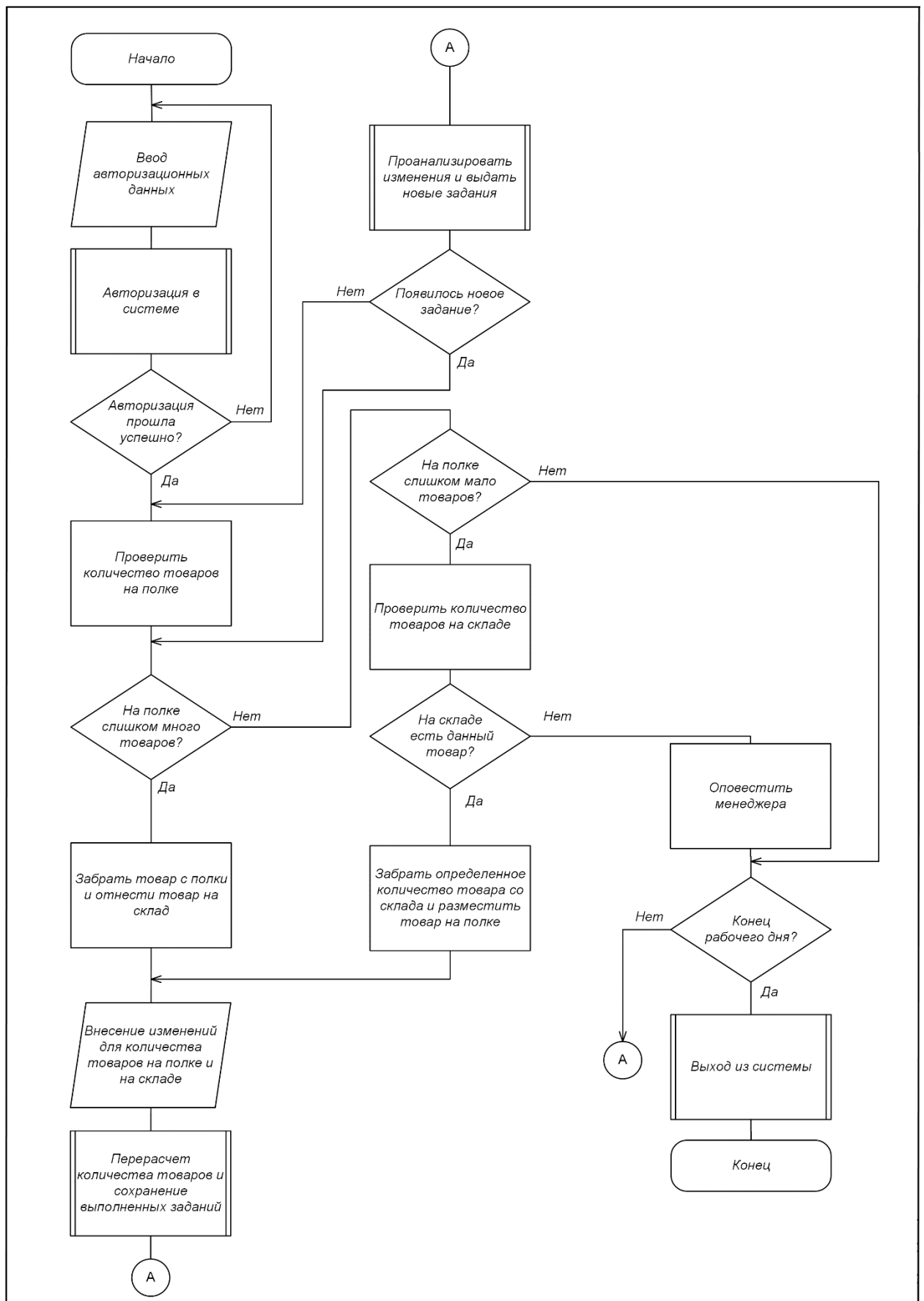


Рисунок 2.7 – Блок-схема функционирования автоматизированной системы



## 2.5 Техническое и системное программное обеспечение

Автоматизированная система имеет архитектуру клиент-сервер. Это значит, что код клиентской части будет изолирован от кода серверной части, а следовательно, исполнение программ будет выполняться на разных устройствах.

Под клиентской частью системы подразумевается веб-приложение, работающее в браузере на устройстве пользователя. Сотрудник супермаркета может использовать любое переносное устройство, поддерживающее браузер и выход в интернет. Минимальные системные требования для устройства пользователя:

- операционная система *Android* или *iOS*;
- процессор *Qualcomm MSM7227T* 800 МГц;
- ОЗУ емкостью 278 Мб;
- наличие выхода в Интернет;
- дисплей с разрешением 320x480 пикселей;
- кнопочная клавиатура либо сенсорное устройство ввода.

Автоматизированная система рассчитана на работу в виде облачного сервиса, что позволит сократить накладные затраты на содержание сервера и масштабировать аппаратную часть в случае необходимости. Поэтому системные требования для отдельно выделенного сервера рассматриваются как необходимые для запуска и начала работы с системой. Таким образом, минимальные системные требования для сервера:

- операционная система *Windows 7*;
- процессор *Intel® Core™ i5–3320M* 2.60 ГГц;
- ОЗУ емкостью от 512 Мб;
- наличие выхода в Интернет;
- встроенная видеокарта;
- клавиатура, мышь.

## 2.6 Эргономическое обеспечение

Эргономика программного обеспечения – это степень, в которой продукт может быть использован пользователем для достижения поставленных целей эффективно и с удовольствием в заданном контексте использования. Основной целью эргономики является сделать труд и отдых наиболее комфортными и максимально продуктивными. Под эргономикой программного обеспечения в большинстве случаев понимают такие вещи, как:

- удобство интерфейса;
- доступность и логичность размещения элементов управления;
- оптимальные цветовые решения.

Эргономичность автоматизированной системы прослеживается в таких областях, как размещение элементов на странице, примерно одинаковые отступы элементов, выбор шрифта, соответствующего контексту, цветовая гамма веб-приложения. Отличительной особенностью системы в плане удобства является адаптивный интерфейс, с помощью которого появляется возможность комфортной работы как через мобильное устройство, так и через персональный компьютер. Также выделяется блочное расположение элементов, благодаря чему пользователь с легкостью может выделить взаимодействующие между собой элементы.

Для оформления страниц был применен стиль расположения основных элементов по центру, что позволяет быстро ориентироваться в рабочей области веб-приложения. Цвет основной области выбран белым, цвет вне основной области – ярко-серый. Цвета выбраны исходя из общепринятых концепций оформления страниц: белый цвет позволяет выделить основные элементы, тогда как серый создает эффект «приглушения». Для выделения названий некоторых элементов применяется полужирный шрифт. Цвета кнопок действий выбраны в соответствии с интуитивно понятными вариантами развития событий при нажатии на них: для перехода на другой элемент – синий, для добавления или изменения элемента – зеленый, для удаления или отмены – красный.

Для оповещений о неполадках выбран светло-красный цвет в связи с тем, чтобы у пользователя не появлялась паника при возникновении внезапной ошибки.

Ключевым аспектом эргономики является отсутствие каких-либо раздражителей либо вещей, которые могут вызвать недовольство либо дискомфорт у пользователя. Чтобы достичь этого, были убраны любые моргающие эффекты, которые могут отвлекать пользователя, отсутствует банерная реклама.

## **2.7 Организационное обеспечение**

Организационное обеспечение – это совокупность методов и средств, регламентирующих взаимодействие работников с техническими средствами и между собой в процессе разработки и эксплуатации программного комплекса.

В данном дипломном проекте к средствам, регламентирующим взаимодействие пользователей с системой будут являться права пользователя. Предполагается наличие четырех видов пользователей, имеющих различные права:

- неавторизованные пользователи;
- персонал;
- менеджер;
- системный администратор.

Тип пользователя зависит в первую очередь от того, авторизован пользователь или нет, а также от прав, предоставляемых пользователю для взаимодействия с системой.

В случае неавторизованного пользователя, доступные функции ограничиваются просмотром некоторых информационных страниц и возможностью авторизации.

При авторизации, базовый набор функций взаимодействия с системой увеличивается. Для персонала данный набор будет включать изменение количества продуктов и просмотр личного кабинета. Кроме того, персоналу также доступны оповещения системы и аналитические данные.

Расширенный функционал имеют менеджер супермаркета и системный администратор. Менеджеру добавляются возможности полного редактирования параметров продуктов, магазина и склада. Также, менеджеру доступна функция редактирования некоторых аналитических параметров и параметров других пользователей.

Системный администратор имеет наиболее полный список доступных функций системы. Данный вид пользователя обладает полномочиями выполнять все действия, которые разрешены другим пользователям. Отличительной особенностью возможностей системного администратора является функция создания новых пользователей в системе. Также данному виду пользователей доступна страница с административной частью системы, на которой есть возможность изменять группы пользователей и самих пользователей.

## 3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ

### 3.1 Обоснование выбора средств разработки

Для разработки и запуска системы необходимо выбрать средства, которые позволят создать востребованный продукт. От выбранных средств разработки будет зависеть как скорость разработки и тестирования, так и качество конечного продукта: от выбранной системы управления базами данных будет зависеть скорость обработки *SQL*-запросов, от выбранной платформы и языка программирования будет зависеть как скорость выполнения задач, так и обновляемость системы, а также ее дальнейшая поддержка.

#### 3.1.1 Выбор системы управления базами данных

При сравнении различных популярных баз данных, следует учитывать, удобна ли для пользователя и масштабируема ли данная конкретная система управления базами данных, а также убедиться, что она будет хорошо интегрироваться с другими продуктами, которые уже используются. Кроме того, во время выбора следует принять во внимание стоимость системы и ее поддержки.

*Microsoft SQL Server* – это система управления базами данных, основная часть которой работает на облачных серверах, а также локальных серверах. Последняя версия *Microsoft SQL Server* поддерживает динамическую маскировку данных, которая гарантирует, что только авторизованные пользователи будут видеть конфиденциальные данные.

Достоинства *Microsoft SQL Server*:

- продукт очень прост в использовании;
- текущая версия работает быстро и стабильно;
- доступ к визуализации на мобильных устройствах;
- хорошо взаимодействует с другими продуктами *Microsoft*.

Недостатки *Microsoft SQL Server*:

- цена для юридических лиц оказывается неприемлемой для большей части организаций;
- даже при тщательной настройке производительности *SQL Server* способен занять все доступные ресурсы.

*PostgreSQL* является одним из нескольких бесплатных популярных вариантов систем управления базами данных, часто используется для ведения баз данных веб-сайтов. Это была одна из первых разработанных систем управления базами данных, поэтому в настоящее время она хорошо развита, и позволяет пользователям управлять как структурированными, так и неструктурированными данными. Может быть использован на большинстве основных платформ, включая *Linux*. Прекрасно справляется с задачами импорта информации из других типов баз данных с помощью собственного инструментария.

Достоинства *PostgreSQL*:

- является масштабируемым и способен обрабатывать терабайты данных;
- поддерживает формат *json*;
- существует множество predefined функций.

Недостатки *PostgreSQL*:

- разбросанная документация, ответы на некоторые вопросы придется искать в интернете;
- скорость работы может падать во время проведения пакетных операций или выполнения запросов чтения.

*SQLite* – это встраиваемая кроссплатформенная база данных, которая поддерживает полный набор команд *SQL* и имеет открытый исходный код на языке *C*. *SQLite* развивается по принципу «минимальный, но полный набор». Она не поддерживает сложные алгоритмы и продукты, но во многом соответствует *SQL 92*, и вводит некоторые свои особенности, которые очень удобны, но не стандартны для языка *SQL*.

Для *SQLite* не требуется сервер приложений, доступ к базе данных происходит через «подключения», которые открываются через вызов функции *DLL*. Система использует механизмы блокировки доступа к файлу на уровне операционной системы, что может быть плохо, если работа с базой данных ведется через удаленный сервер. Изначально *SQLite* работал по принципу «многие читают — один пишет».

Достоинства *SQLite*:

- распространяется бесплатно;
- общедоступна и распространена на многих платформах и операционных системах;
- программное обеспечение базы данных полностью покрыто автоматическими тестами;

- легко переносить при смене системы или платформы.

Недостатки *SQLite*:

- не подходит для крупных производственных систем;
- нельзя удалить столбец в таблице;
- нет хранимых процедур;
- медленные операции манипуляции данными при большом количестве данных.

Так как для *Microsoft SQL Server* необходимо много ресурсов, а также большая часть возможностей предоставляется по коммерческой подписке, для разрабатываемой системы могут быть использованы *PostgreSQL* или *SQLite*. Учитывая, что для *PostgreSQL* производительность может падать при частом считывании данных, данную систему управления базами данных стоит отклонить. *SQLite* предоставляет достаточно возможностей для взаимодействия с данными, кроме того поддержка *SQLite* реализована на уровне фреймворка *Django*, поэтому выбор становится в пользу *SQLite*.

### 3.1.2 Выбор средств разработки

При сравнении различных языков программирования, следует учитывать стоимость системы и ее последующей поддержки. Важной составляющей при данном выборе будет являться возможность в дальнейшем обновить создаваемую систему для введения последних языковых изменений без каких-либо дополнительных накладок. Кроме того следует учесть возможность быстрого и качественного введения изменений в систему, что позволит развивать ее на этапе поддержки.

*HTML* (*HyperText Markup Language*) – основная составляющая разработки пользовательского интерфейса. С помощью языка гипертекстовой разметки появляется возможность создавать различные элементы на веб-странице, которые будут отображать полезные данные в браузере пользователя. При использовании *HTML* обычно в связке идет *CSS* (*Cascading Style Sheets*) – язык описания внешнего вида *HTML*-документа. Данный язык позволит применить оформление внешнего вида веб-страниц с использованием различных цветов, шрифтов, стилей и расположения отдельных блоков.

*JavaScript* – это язык, который позволяет применять сложные вещи на веб-странице, такие, как отображение периодически обновляемого контента или интерактивных карт, 2D/3D графики, прокрутка видео в проигрывателе и так далее. Изначально его цель состояла лишь в том, чтобы добавить немного

поведения на веб-страницу. Сейчас *JavaScript* продолжает использоваться для создания веб-сайтов, только теперь он предоставляет гораздо больше возможностей. Данный язык программирования будет полезен при разработке пользовательской части системы.

В качестве дополнительной библиотеки для повышения скорости разработки и упрощения взаимодействия между компонентами, к *JavaScript* добавляется библиотека *JQuery*, которая фокусируется на взаимодействии *JavaScript* и *HTML*, а также предоставляет удобные прикладные интерфейсы для работы с *AJAX*, который представляет собой подход к построению пользовательских интерфейсов с использованием асинхронности.

Чтобы избежать создания элементов с нуля, также будет использоваться фреймворк *Bootstrap* – свободный набор инструментов для создания сайтов и веб-приложений. Для отрисовки некоторых элементов на странице пользователя, таких как графиков и диаграмм, используется библиотека *Google Charts*.

В качестве языка, на котором будет написана серверная часть системы, был выбран язык *Python*, который широко применяется как интерпретируемый язык для скриптов различного назначения. Элегантный дизайн и эффективный, дисциплинирующий синтаксис этого языка облегчают программистам совместную работу над кодом. *Python* – мультипарадигмальный язык программирования, который позволяет совмещать процедурный подход к написанию кода с объектно-ориентированным и функциональным.

Для построения взаимодействия серверной части системы с базой данных и клиентской частью, необходимо использовать набор заранее написанных библиотек. В данном случае, в связке с *Python* будет использоваться фреймворк *Django* – популярный и полнофункциональный серверный веб-фреймворк.

При разработке пользовательской части системы, кроме языков *JavaScript*, *HTML* и *CSS* будет использоваться язык шаблонов фреймворка *Django*. Данный язык позволяет строить интерфейс с использованием *back-end* кода, который не отображается на веб-странице, но производит некоторые предварительные вычисления для добавления некоторого дополнительного функционала. Пример использования языка шаблонов – разграничение функционала пользователей. При использовании языка шаблонов достаточно будет прописать несколько условий для отображения определенных данных на странице в зависимости от текущего пользователя, что избавляет от необходимости повторять веб-страницы для каждого пользователя отдельно.

### 3.2 Структура программных средств автоматизированной системы

Перед проектированием сложного программного продукта необходимо определиться с его структурой. Для этого нужно определить структурные компоненты программного продукта, а также связи между ними, а после этого составить структурную схему.

Структурная схема автоматизированной системы управления полочным пространством супермаркета представлена на рисунке 3.1.

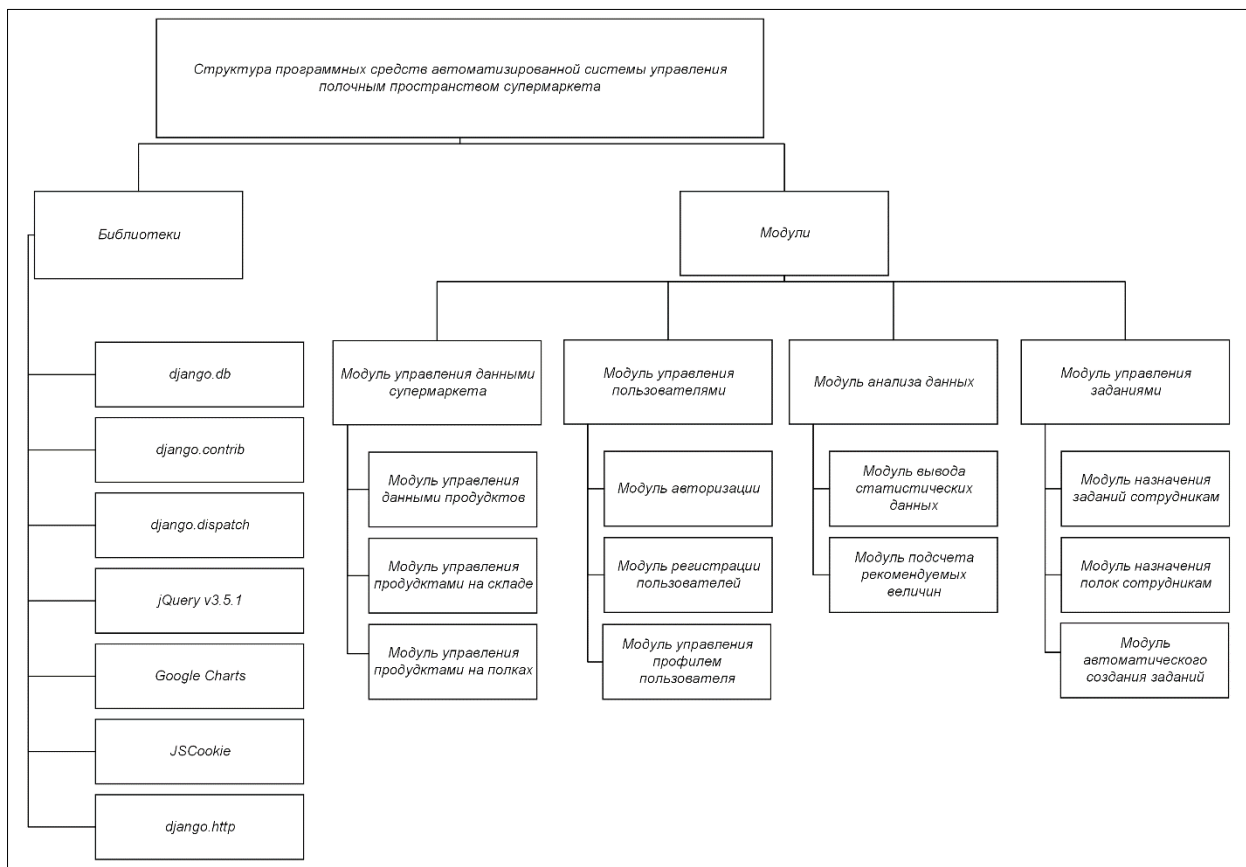


Рисунок 3.1 – Структурная схема автоматизированной системы

Модули разделены между собой в соответствии с решаемой ими задачей. Они могут взаимодействовать друг с другом если функционал одного модуля предоставляет вычисления, необходимые для другого модуля. Как пример такого взаимодействия можно привести вычисление аналитических данных, вычисляемых модулем анализа данных, необходимых для создания заданий модулем управления заданиями.

Модуль управления данными супермаркета контролирует данные о перемещении продуктов между супермаркетом и складом. Кроме того, данный



модуль выполняет контроль продажи продуктов для последующего управления данными о продуктах.

Модуль управления пользователями разграничивает функции, доступные пользователям разных групп. Данный модуль предоставляет функционал для регистрации новых пользователей, последующей авторизации и управления профилем каждого пользователя.

Модуль анализа данных анализирует поступающие данные, в результате чего происходит построение графиков на основе статистических данных. Кроме того, данный модуль вычисляет рекомендуемые параметры работы системы на основе сохраненных ранее статистических данных.

Модуль управления заданиями следит за изменениями количества продуктов, на основе чего принимает решение о создании и назначении заданий пользователям. Также данный модуль позволяет менеджеру назначать ответственных за определенную полку или задачу пользователей.

### **3.3 Разработка программного кода автоматизированной системы**

Так как в качестве фреймворка для разработки данной автоматизированной системы был взят *Django*, основная часть программного кода будет реализована на шаблоне веб-проектирования *MVC* – модель-представление-контроллер. Данный шаблон подразумевает разделение всего приложения на три компонента: модель данных приложения, пользовательский интерфейс и логику взаимодействия пользователя с системой. Такой подход позволяет агрегировать логику отдельно взятых компонентов, благодаря чему изменение одного из них минимально сказывается на остальных компонентах.

Модель автоматизированной системы предоставляет данные о ее компонентах и их взаимодействиях в процессе работы, которые хранятся в базе данных. Связь с базой данных в *Django* осуществляется с помощью технологии *ORM* – объектно-реляционного отображения. Данная технология позволяет связать объекто-ориентированный подход с базой данных, в результате чего создаются классы моделей для каждой сущности. С помощью данных классов контроллеры могут взаимодействовать с объектами базы данных как если бы это были обычные экземпляры классов. Классы моделей прописываются в файле *models.py*.

Представление отвечает за отображение информации пользователю. Данный компонент оборачивает данные, полученные из модели, в *HTML* и *CSS* код с использованием *JavaScript*. Благодаря представлениям одни и те же

данные можно представить в различном виде для разных пользователей, будь то другой язык или разная функциональность для отдельных групп пользователей. Данные для отображения приходят в представления через запросы передачи состояния представления, обернутые в специальные адаптеры на языке *Python*. Адреса для отображения представлений настраиваются отдельно, весь основной код представлений расположен в файле *views.py*.

Контроллер обеспечивает возможность взаимодействия пользователя с системой. Данный компонент связывает представление и модель для предоставления определенной функциональности системы при изменении данных в связанных компонентах. Контроллеры предоставляют основные возможности системы. На уровне контроллера происходит анализ и фильтрация данных, проверяются права пользователя при авторизации и переходе по страницам, выполняются необходимые расчеты и другие действия. В связи с модульностью системы, данный компонент разделен на отдельные модули, каждый из которых предоставляет функционал для определенной части системы.

Кроме использования паттерна «модель-представление-контроллер» *Python* дает возможность реализовывать код в функциональном стиле, что позволяет разрабатывать масштабируемые модули системы. Данный факт означает, что многие функции могут быть реализованы таким образом, чтобы код выполнялся без каких-либо значимых побочных эффектов, что в свою очередь позволяет заранее знать поведение той или иной функции. Также данный подход позволит реализовывать некоторые общие функции, которые выполняют одни и те же действия для разных представлений и контроллеров, и эти функции в последствии могут быть преобразованы в какие-либо другие функции без потери исходного смысла.

Для реализации некоторых возможностей, таких как построение графиков и диаграмм, работа с датами, стандартных математических функций, используются готовые библиотеки, которые подключаются к файлам системы в процессе разработки.

### 3.4 Описание программной реализации отдельных частей автоматизированной системы

#### 3.4.1 Модуль управления данными супермаркета

Данный модуль предоставляет функционал для управления данными товаров, которые находятся на складе и на полках супермаркета.

Для управления параметрами продуктов супермаркета используется компонент *products.py*. Данная возможность предоставляется только менеджеру супермаркета и системному администратору. Кроме изменения существующих продуктов возможно добавление новых и удаление старых:

```
def removeProduct(payload):
    productId = payload['product_id']
    removedProduct = models.Product.objects.get(id=productId)
    removedProduct.delete()
```

При сохранении измененных параметров происходит проверка на допустимость введенных значений. В случае неудачи изменения отклоняются, пользователю приходит уведомление об этом:

```
def saveChanges(payload):
    isDigitCodeValid = bool(re.match("[0-9]{5}-[0-9]{4}$", newDigitCode))
    if not isDigitCodeValid:
        result['failure'] = "неправильный формат кода"
    else:
        try:
            hasDuplicates = models.Product.objects.filter(
                ~Q(id=productId)).get(digit_code=newDigitCode)
        except models.Product.DoesNotExist:
            hasDuplicates = False
        if hasDuplicates:
            result['failure'] = "код должен быть уникальным"
    product.save()
```

*storage.py* – основной компонент управления складом, предоставляет персоналу возможность добавления некоторого количества продуктов на склад. Информация о продуктах берется из ранее добавленных с помощью компонента *products.py* данных. При добавлении продуктов изменяется параметр *delivery\_timestamp*, который хранит дату последней поставки продукта на склад. Менеджеру позволяет изменять количество товаров на складе:

```
def saveProductsCount(payload):
    productsCount = int(payload['productsCount'])
    storageElem.product_count = productsCount
    storageElem.save()
```

Компонент управления полками супермаркета содержит функционал изменения количества продуктов на полке. Персонал имеет возможность добавлять некоторое количество продуктов, при этом изменяется параметр *last\_charge*, который имеет смысл даты и времени последней загрузки полки. При изменении количества продуктов на полке выполняется функция *checkForFailures* компонента *failures\_manager*, о котором будет говориться позже:

```
def addProducts(payload):
    newProductCount = storeElem.product_count + addCount
    if newProductCount > 0:
        storeElem.product_count += addCount
        storeElem.last_charge = datetime.datetime.now()
        storeElem.save()
        failures_manager.checkForFailures(storeElemId)
    else:
        result['failure'] = "количество забираемых элементов не может превышать количество элементов на складе"
```

Менеджеру доступны функции добавления и удаления секций и полок, изменение параметров полок:

```
def removeShelf(payload):
    shelfElemId = payload['shelfElem_id']
    removedShelf = models.Store.objects.get(id=shelfElemId)
    removedShelf.delete()
```

Для внутренних задач системы реализован отдельный компонент, который управляет данными базы данных на прямую. Данный компонент предоставляет функции ввода данных для каждого из компонентов системы, а также функцию восстановления базы данных в определенных случаях:

```
def restore_db():
    models.Product.objects.all().delete()
    models.Storage.objects.all().delete()
    models.Store.objects.all().delete()
    models.Stock.objects.all().delete()
    models.Statistics.objects.all().delete()
```

```
fill_in_products()
fill_in_storage()
fill_in_store()
fill_in_stocks()
fill_in_statistics()
```

### 3.4.2 Модуль управления пользователями

Модуль управления пользователями предоставляет возможности регистрации новых пользователей системным администратором, авторизации пользователей и управления профилем авторизованного пользователя. Основной функционал регистрации, авторизации и аутентификации реализован стандартными библиотеками *Django*. Регистрировать пользователей имеет возможность только системный администратор.

Для управления пользовательским профилем предназначен компонент *profile.py*. Данный компонент реализует возможности изменения данных каждого пользователя и управления заданиями, за которые будет ответственен пользователь. Назначать и забирать задания могут только менеджер и системный администратор:

```
def assignTask(payload):
    task = models.Failure.objects.get(id=taskId)
    if assigneeId:
        task.assignee = models.User.objects.get(id=assigneeId)
    else:
        task.assignee = None
    task.save()
```

Пользователю предоставляется возможность просматривать доступные для него полки, просматривать невыполненные задания, а также ставить их выполнение с помощью функции *solveTask*:

```
def solveTask(payload):
    taskId = payload["task_id"]
    task = models.Failure.objects.get(id=taskId)
    task.is_solved = True
    task.save()
```

Кроме просмотра и управления заданиями, пользователи имеют возможность просматривать полки, на которые они назначены.

### 3.4.3 Модуль анализа данных

Модуль анализа данных представлен компонентом *analytics.py* и предоставляет возможности анализа вводимых в систему данных. Данный модуль вычисляет рекомендуемые параметры на очередной день работы супермаркета. Для вычисления рекомендуемых параметров прежде всего рассчитываются максимально и минимально допустимые количества товаров. Для дальнейших расчетов берутся данные за несколько предыдущих дней работы. По определенным правилам рассчитывается средний показатель максимальной и минимальной востребованности товара на определенной полке. Далее берутся данные за несколько прошлых лет работы супермаркета, соответствующие примерному дню и месяцу, для которого происходит расчет. Также по определенным правилам рассчитываются наибольшая и наименьшая востребованность товаров в те дни. После вычислений данных показателей, происходит вычисление их среднего значения. При отсутствии показателей за определенный период времени, он пропускается и не учитывается в подсчетах:

```
def getStatsForShelf(shelf):
    min_products_count = math.floor(shelf.width / shelf.product.width)
    if shelf.product.stackable:
        max_products_count = math.floor(shelf_V / product_V)
    else:
        max_products_count = math.floor(shelf_S / product_S)
    if min_products_count > max_products_count:
        min_products_count = max_products_count
    products_count_average = (max_products_count + min_products_count) / 2
    min_recommended = math.floor(min_for_day_sum / stats_count)
    max_recommended = math.ceil(max_for_day_sum / stats_count)
    calculatedStats = calculateLastYearStats(
        currentDate, shelf, min_products_count, max_products_count,
        years_to_count, 365
    )
    calculatedStats = calculateLastYearStats(
        currentDate, shelf, min_products_count, max_products_count,
        years_to_count, 365 * 2
    )
    min_final = math.ceil((min_products_count + min_recommended +
        min_year_ago + min_two_years_ago) / years_to_count)
    max_final = math.floor((max_products_count + max_recommended +
        max_year_ago + max_two_years_ago) / years_to_count)
    avg_recommended = math.ceil(
        ((min_final + max_final)/2 + products_count_average)/2)
```

Данная операция производится для каждой полки, в результате чего высчитываются рекомендованные параметры, которые учитываются модулем управления заданиями и выводятся на странице *analytics*.

Также данный модуль позволяет заносить новые данные на определенный день с проверкой их правильности. Эта возможность доступна только менеджеру и системному администратору. Кроме того, модуль анализа данных выдает статистические данные на каждую полку. Данные рассчитываются для нескольких лет и выводятся в виде графиков и диаграмм на странице *analytics*:

```
def getFullStats():
    storeShelves = models.Store.objects.all()
    fullStatsRaw = models.Statistics.objects.order_by('day')
    years = list(
        map(
            lambda singleYearRaw: singleYearRaw.strftime('%Y'),
            models.Statistics.objects.all().dates('day', 'year')
        )
    )
    if soldInYear:
        availableStats["soldEveryDay"] = soldInYear
    if incomeInYear:
        availableStats["incomeEveryDay"] = incomeInYear
    if failuresInYear:
        availableStats["failuresInYear"] = failuresInYear
    fullStatsForEachShelf.append(
        [storeShelf, availableStats]
    )
```

Полный листинг кода модуля анализа данных представлен в приложении А.

#### 3.4.4 Модуль управления заданиями

Модуль управления заданиями продолжает функциональность компонента управления профилями модуля управления пользователями. Данный модуль реализует распределение заданий между пользователями системы, а также автоматически создает новые задания в случаях, когда количество товаров на полках зайдет за границы допустимых рекомендованных значений.

Каждый пользователь может посмотреть на свои текущие и решенные задания на странице *failures*. Пользователю доступна возможность пометить

задание как решенное, однако система при очередном пополнении полки товарами проверит выполнение задания, и в случае, если рекомендации системы не были выполнены, снова создаст задание с похожим описанием. Кроме того, если пользователь выполнит рекомендации системы, она автоматически пометит задание как выполненное:

```
def checkForFailures(shelfId):
    if shelfAlreadyInFailures and not failure.is_solved:
        break
    else:
        shelfAlreadyInFailures = False
    if min_recommended > shelfElem.product_count and not shelfAlreadyInFailures:
        models.Failure.objects.create
    elif max_recommended < shelfElem.product_count and not shelfAlreadyInFailures:
        models.Failure.objects.create
    elif shouldBeChecked and not shelfAlreadyInFailures:
        models.Failure.objects.create
    if shelfAlreadyInFailures:
        for failure in models.Failure.objects.all():
            if bool(re.match(shelfElem.shelf_name, failure.text)) and bool(
                re.match(shelfElem.section_name, failure.text)):
                if min_recommended < shelfElem.product_count and
max_recommended > shelfElem.product_count and not shouldBeChecked:
                    failure.is_solved = True
                    failure.save()
```

Кроме управления непосредственно заданиями, данный модуль предоставляет функционал управления полками. Менеджеру и системному администратору доступна функция назначения и отмены назначения полок пользователям:

```
def assignShelf(payload):
    storeElemId = payload['shelf_id']
    storeElem = models.Store.objects.get(id=storeElemId)
    profile = None
    if profileId:
        profile = models.User.objects.get(id=profileId)
    storeElem.user = profile
    storeElem.save()
```



### 3.5 Руководство пользователя

Работа с автоматизированной системой начинается со входа на домашнюю страницу системы. Неавторизованный пользователь получает сообщение о том, что ему следует авторизоваться в системе для продолжения работы с ней. Изначально доступны минимальный набор страниц и базовая функциональность. Главная страница для неавторизованного пользователя представлена на рисунке 3.2.

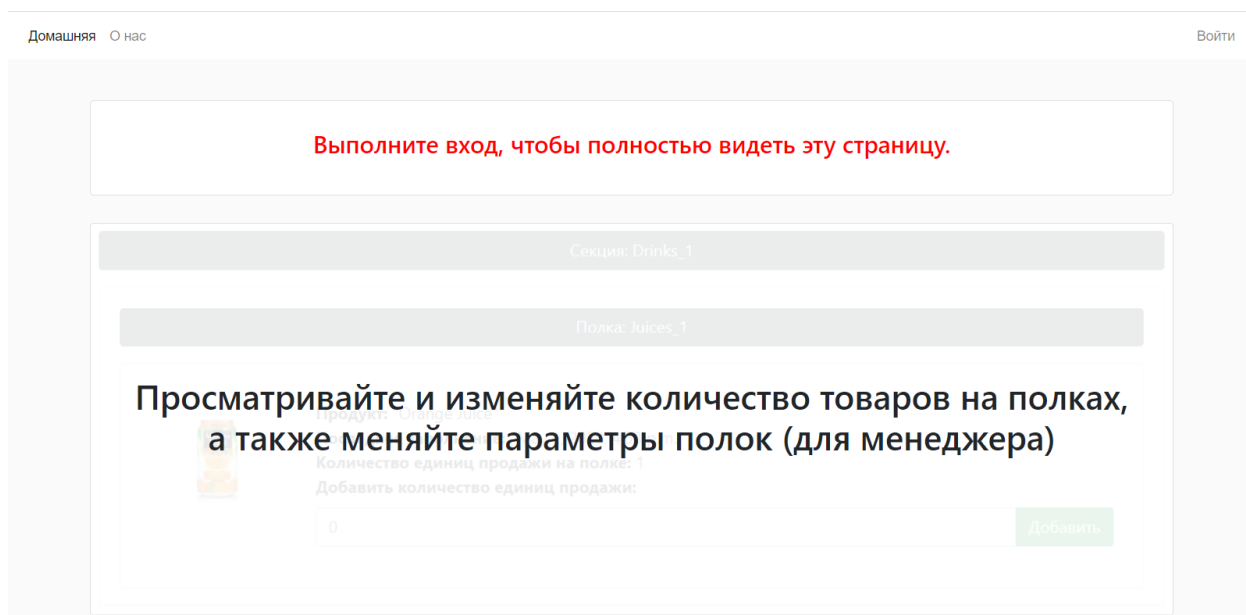


Рисунок 3.2 – Главная страница системы для неавторизованного пользователя

На стартовой странице пользователь может увидеть возможности, которые предоставляет система, а также перейти на страницы с контактной информацией владельца системы, нажав на кнопку «О нас», и авторизации при нажатии на кнопку «Войти».

При нажатии на кнопку «О нас» пользователь переходит на страницу, где ему предоставляется контактная информация владельца системы. Данная страница изображена на рисунке 3.3.

При переходе на страницу авторизации пользователю выдается форма для ввода его авторизационных данных. Если пользователь введет неправильные данные, система предупредит его об этом сообщением над формой. Страница входа в систему изображена на рисунке 3.4

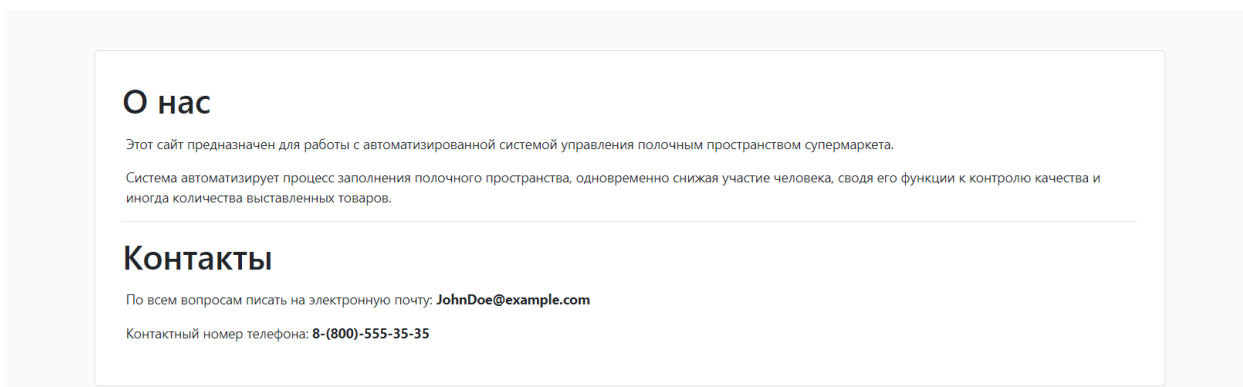


Рисунок 3.3 – Страница «О нас»

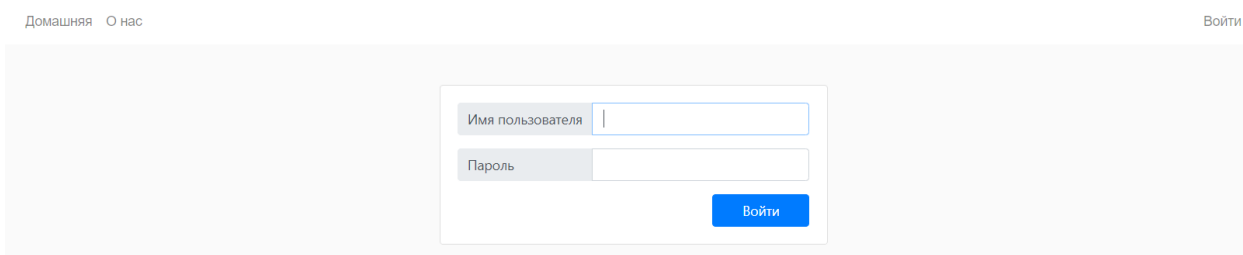


Рисунок 3.4 – Страница ввода авторизационных данных

### 3.5.1 Руководство пользователя для персонала

При вводе правильных авторизационных данных система переносит пользователя на главную страницу сайта. Авторизованному пользователю становятся доступны новые страницы и функции системы. Главная страница сайта в режиме «Персонал» изображена на рисунке 3.5.

Для авторизованного пользователя отображаются новые кнопки, недоступные без авторизации. Данный набор страниц предоставляет возможность работы с системой в режиме персонала с определенным набором функций, ограниченных правами пользователя. Однако это не полный набор доступных ссылок на другие страницы и возможности системы. Кроме кнопок на главной странице, появляется выпадающий список из ссылок на другие страницы системы. Список доступных страниц в выпадающем списке изображен на рисунке 3.6.

Также к заголовочному меню сайта добавляется приветствие с именем авторизованного пользователя. При нажатии на имя, пользователь попадает в

личный кабинет. Приветствие со ссылкой на личный кабинет изображены на рисунке 3.7.

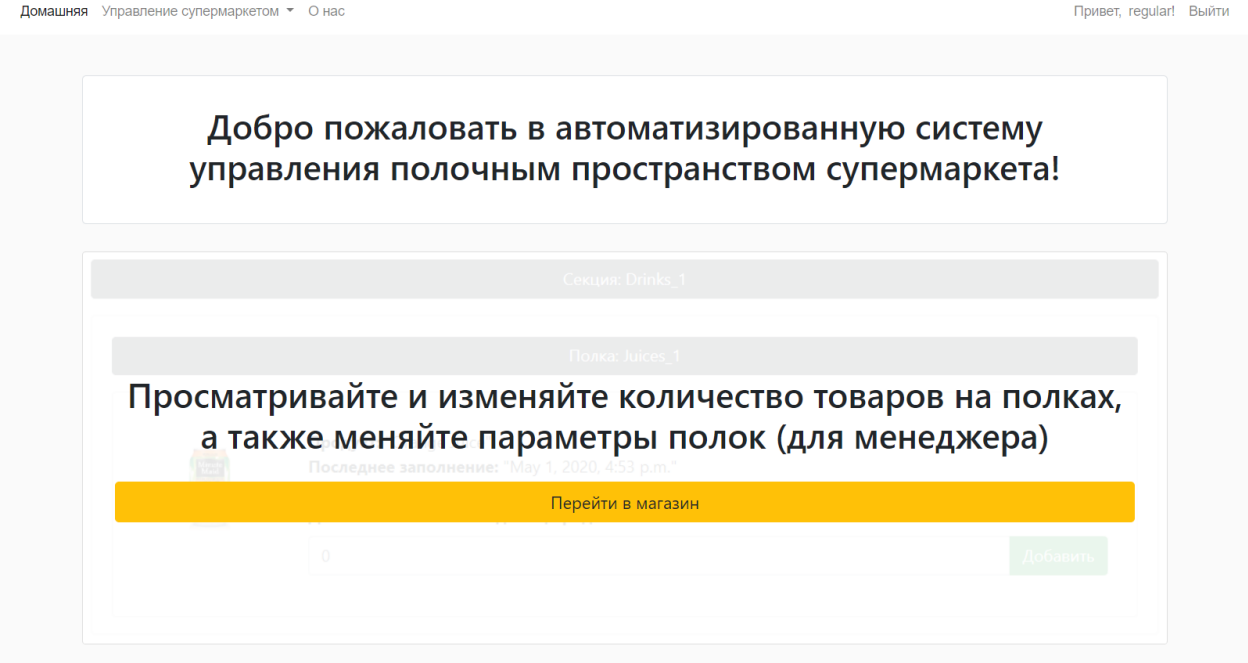


Рисунок 3.5 – Главная страница системы для пользователя в режиме персонала

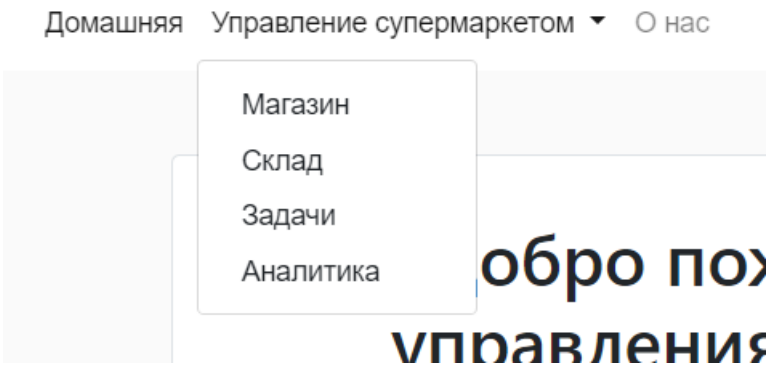


Рисунок 3.6 – Выпадающий список доступных страниц



Рисунок 3.7 – Приветствие со ссылкой на личный кабинет

При переходе в личный кабинет, пользователь видит собственную информацию, которую может редактировать. Также ему отображается список нерешенных задач и назначенных на него полок с краткой информацией о том, что на данной полке расположено и в каком количестве. Личный кабинет пользователя изображен на рисунке 3.8.

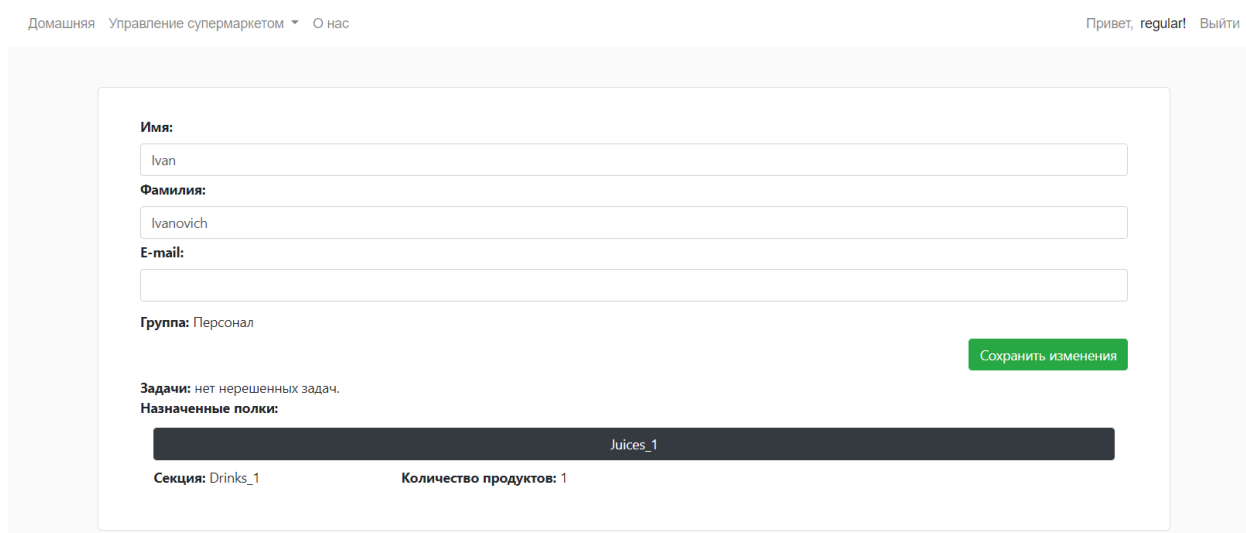


Рисунок 3.8 – Личный кабинет пользователя в режиме персонала

Авторизованному пользователю в режиме «Персонал» также доступна страница «Склад», на которой выдается информация о продуктах, находящихся на складе, их количестве и дате последней поставки. Пользователь в данном режиме может менять количество продуктов на складе. Страница «Склад» изображена на рисунке 3.9.

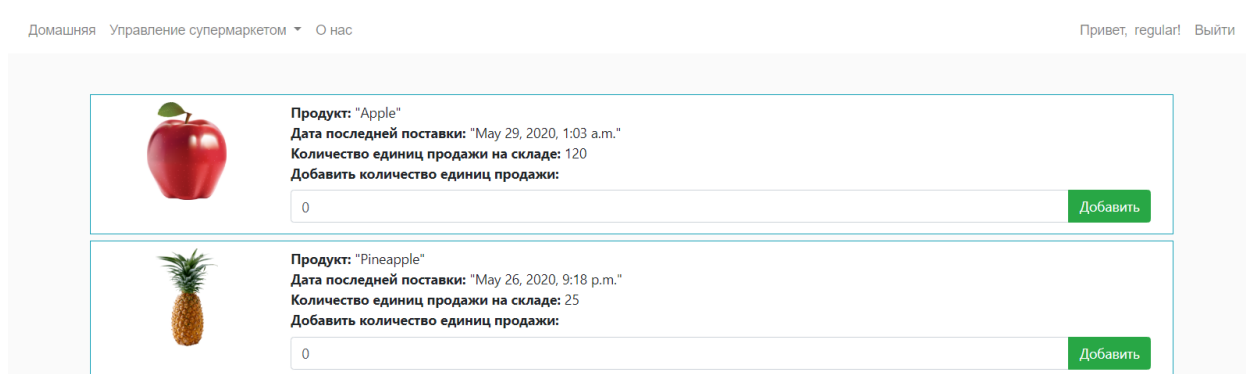


Рисунок 3.9 – Страница «Склад» в режиме персонала

Также персоналу доступна страница магазина, на которой отображаются кнопки с названиями секций. При нажатии на кнопку секции разворачивается список полок, относящихся к данной секции. При нажатии на кнопку полки, относящейся к определенной секции, выдается информация по данной полке: товар, который находится на полке, последнее заполнение полки, количество единиц продажи на полке в данный момент времени. Персонал имеет возможность добавлять или отнимать некоторое количество товара, что изменяет его количество на полке. Страница магазина изображена на рисунке 3.10.

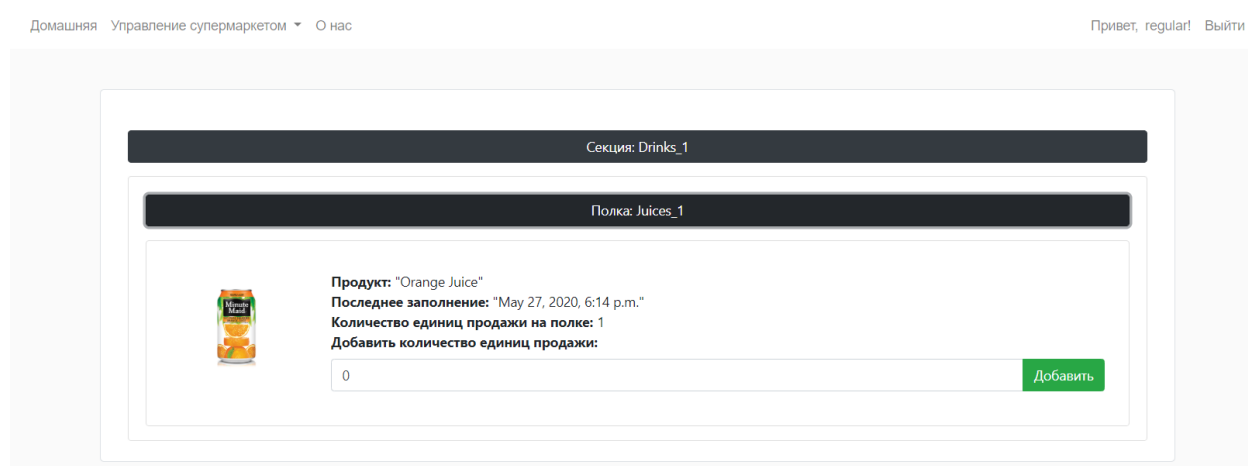


Рисунок 3.10 – Страница «Магазин» в режиме персонала

Авторизованный пользователь имеет возможность просматривать список своих решенных и нерешенных задач на странице «Задачи». На данной странице пользователю предоставляется описание ошибки в виде кнопки, при нажатии на которую он может посмотреть возможную причину возникновения задачи, возможное решение, дату возникновения, серьезность ошибки, в результате которой произошло создание задания, статус решения задачи и фамилию с именем пользователя, которому было поручено заняться решением возникшей проблемы. Кроме того, пользователь может отметить задачу как решенную, что переведет ее в статус «решена» и перенесет в список решенных задач. Страница с задачами пользователя изображена на рисунке 3.11.

Последняя доступная персоналу страница – «Аналитика». На этой странице пользователю предоставляется информация по всем полкам: статистика продаж за последние три года, динамика ежедневного дохода, годовые соотношения созданных задач, а также рекомендуемые граничные и

среднее количество товаров на полке на текущий день. Страница с аналитическими данными изображена на рисунке 3.12.

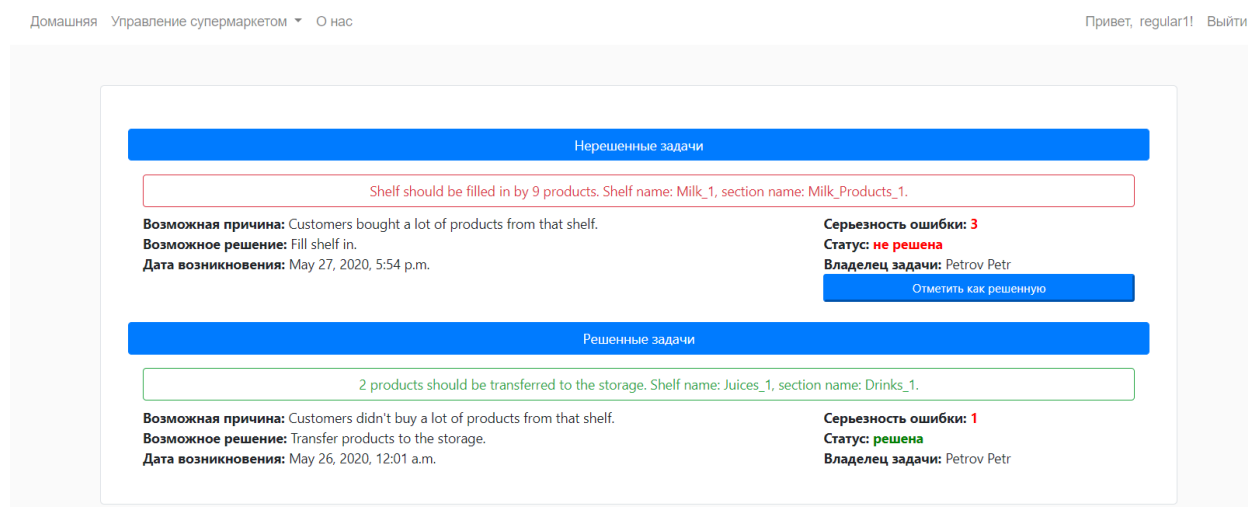


Рисунок 3.11 – Страница «Задачи» в режиме персонала



Рисунок 3.12 – Страница «Аналитика» в режиме персонала

По окончании работы с системой пользователь выходит из своей учетной записи по нажатию на кнопку «Выйти».

### 3.5.2 Руководство пользователя для менеджера и системного администратора

При входе в систему под учетной записью менеджера или системного администратора, пользователю доступны все возможности, которые предоставлялись пользователю в режиме «Персонал».

Первой дополнительной возможностью становится возможность перехода на страницу редактирования информации о продуктах. На главной странице добавляется новая кнопка перехода, изображенная на рисунке 3.13.

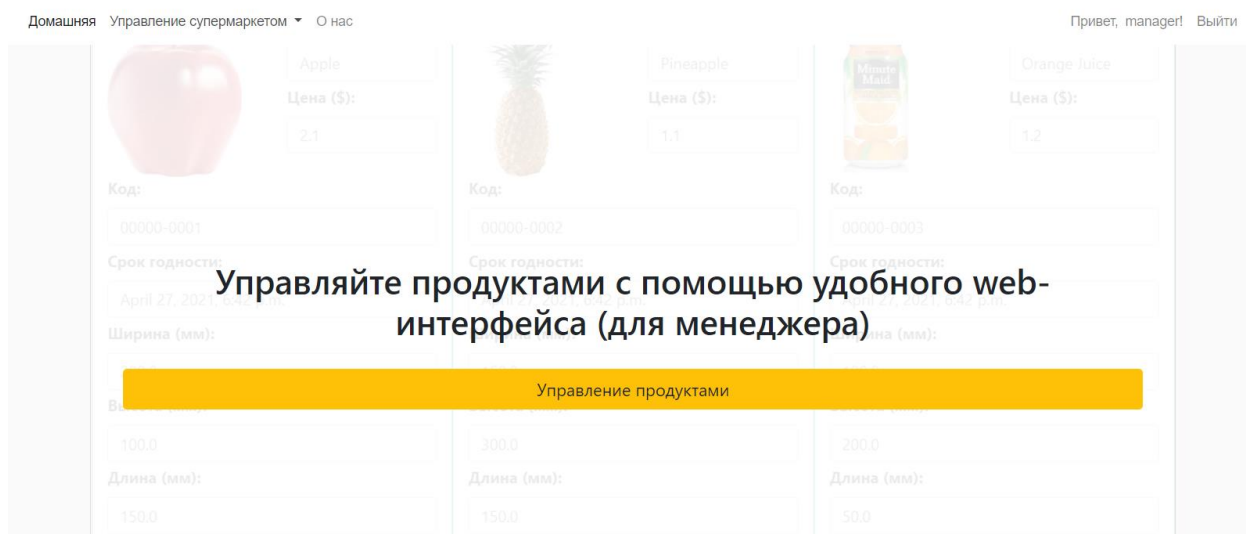


Рисунок 3.13 – Кнопка перехода к управлению продуктами на главной странице в режиме менеджера и системного администратора

Также данный переход выполнен в виде ссылки в выпадающем списке, который можно найти в заголовочном меню сайта. Список с новой ссылкой изображен на рисунке 3.14.

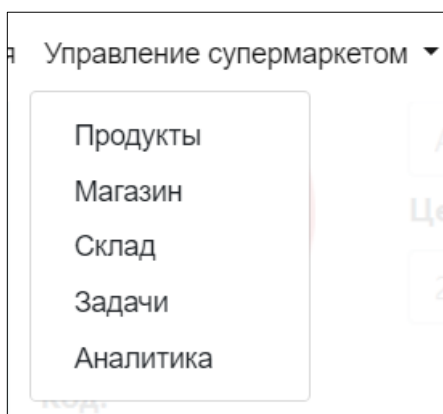


Рисунок 3.14 – Выпадающий список доступных страниц в режиме менеджера и системного администратора

Пользователь под учетной записью менеджера или системного администратора имеет полный контроль над создающимися заданиями. Он

может просматривать их, назначать на определенного пользователя, принадлежащего группе «Персонал», переназначать и отмечать как выполненные. При переходе в личный кабинет менеджеру и системному администратору выводится список еще не выполненных заданий, назначенных на определенного пользователя либо ждущих назначения. Список заданий в личном кабинете пользователя в режиме менеджера и системного администратора изображен на рисунке 3.15.

Домашняя Управление супермаркетом О нас Привет, manager! Выйти

**Группа:**  
Менеджеры Сохранить изменения

**Распределение заданий:**

**Нераспределенные задачи:**

- Shelf should be filled in by 10 products. Shelf name: Potatoes\_1, section name: Vegetables\_1.
- Shelf should be filled in by 5 products. Shelf name: Cucumbers\_1, section name: Vegetables\_1.
- Shelf should be filled in by 5 products. Shelf name: Milk\_2, section name: Milk\_Products\_1.

**Возможная причина:** A lot of time has passed since the last update.  
**Возможное решение:** Clarify the number of products.  
**Дата возникновения:** May 26, 2020, 12:01 a.m.

**Серьезность ошибки:** 3  
**Статус:** не решена

Назначить задачу

**Задачи сотрудника Petrov Petr:**

- Shelf should be filled in by 9 products. Shelf name: Milk\_1, section name: M

**Задачи сотрудника Sergeev Sergey:**

- Number of products should be clarified. Shelf name: Potatoes\_1, section name

**Задачи сотрудника Hanks Tom:**

Ivanovich Ivan  
 Petrov Petr  
 Sergeev Sergey  
 Toster Alex  
 Starz Gregory  
 Hanks Tom  
 Отметить как решенную

Рисунок 3.15 – Список невыполненных заданий в личном кабинете пользователя в режиме менеджера и системного администратора

Кроме тех функций, которые доступны пользователю в личном кабинете в режиме персонала, менеджеру и системному администратору добавляется возможность назначать ответственность за полки на персонал с кратким описанием самой полки, что изображено на рисунке 3.16.

Возможность просматривать и назначать решенные задания на каждого пользователя, который относится к группе «Персонал», продублирована на странице задач. Менеджеру и системному администратору также дается возможность просматривать все решенные и нерешенные задания. Страница заданий для пользователя в режиме менеджера и системного администратора изображена на рисунке 3.17.

Как и персоналу, менеджеру доступна функция просмотра всех продуктов, находящихся на складе, а также добавления их некоторого количества. Также добавляется возможность изменять количество продуктов



напрямую. Страница склада для пользователя в режиме менеджера и системного администратора изображена на рисунке 3.18.

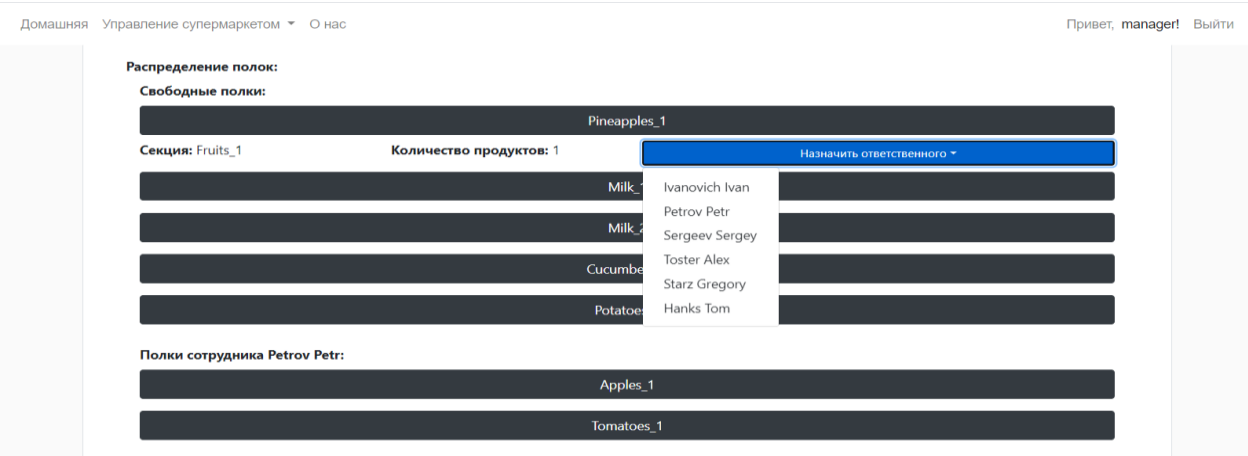


Рисунок 3.16 – Список полок для пользователя в режиме менеджера и системного администратора

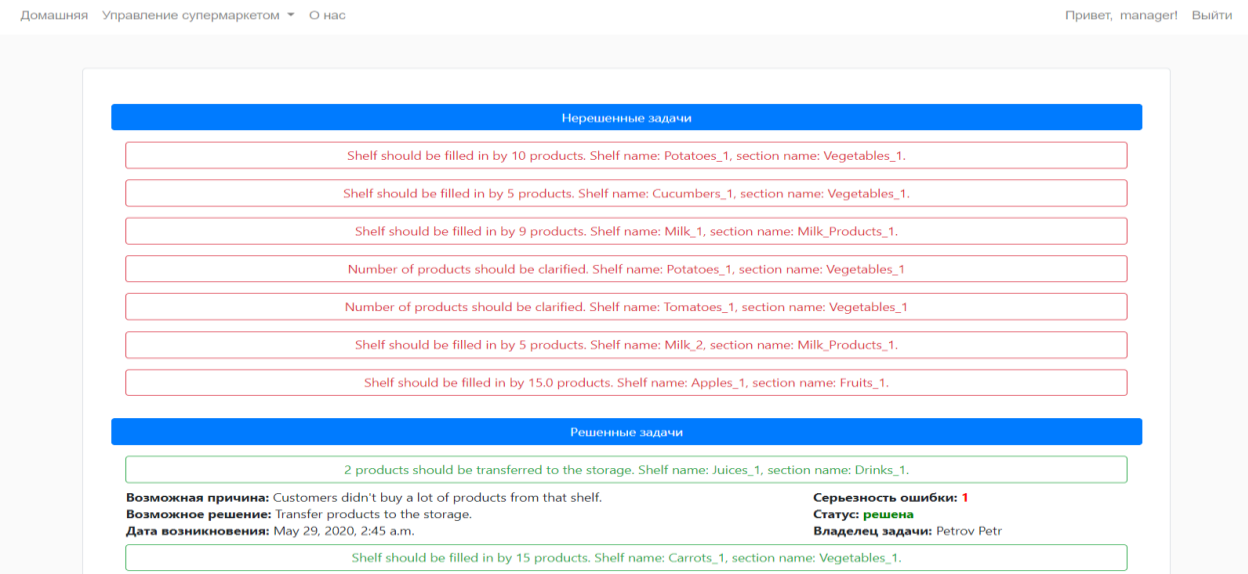


Рисунок 3.17 – Список заданий для пользователя в режиме менеджера и системного администратора

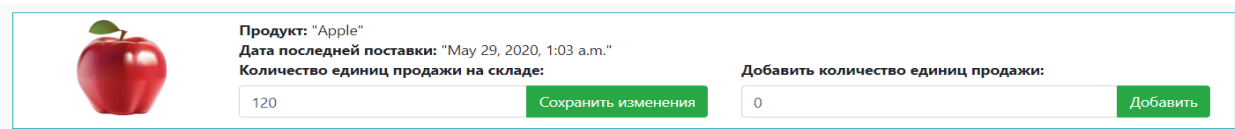


Рисунок 3.18 – Дополнительные возможности на странице «Склад» для пользователя в режиме менеджера и системного администратора

Пользователю в роли системного администратора или менеджера доступен полный контроль над полками супермаркета в системе. Данные группы пользователей имеют возможность добавлять новые секции, редактировать их название, удалять секции, добавлять полки в каждую из секций, и также полностью редактировать параметры каждой полки. Страница магазина для групп пользователей «Системный администратор» и «Менеджер» изображена на рисунке 3.19.

Добавить секцию

Название секции: Fruits\_1 Сохранить

Секция: Fruits\_1

Удалить секцию

Добавить полку

Название полки: Apples\_1 Сохранить

Полка: Apples\_1

Параметры полки Apples\_1:

Ширина (мм): 1000.0 Сохранить Высота (мм): 400.0 Сохранить

Длина (мм): 500.0 Сохранить Макс. вес (г): 100000.0 Сохранить

Удалить полку

Продукт на полке: Apple Поменять продукт

Рисунок 3.19 – Страница магазина для пользователей в режиме менеджера и системного администратора

Пользователи, относящиеся к группам «Системный администратор» и «Менеджер» наряду с персоналом имеют возможность просматривать аналитические данные на странице аналитики. Также им становится доступна функция добавления новых статистических данных, что изображено на рисунке 3.20.

Добавить данные:

Дата:

Количество проданных товаров:

Идентификатор акции (опционально):

Количество ошибок:

Цена продукта в этот день:

Добавить

Рисунок 3.20 – Дополнительные возможности на странице «Аналитика» для пользователей в режиме менеджера или системного администратора

## **4 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБСНОВАНИЕ РАЗРАБОТКИ И ИСПОЛЬЗОВАНИЯ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ УПРАВЛЕНИЯ ПОЛОЧНЫМ ПРОСТРАНСТВОМ СУПЕРМАРКЕТА**

### **4.1 Характеристика автоматизированной системы управления полочным пространством супермаркета**

Автоматизированная система управления полочным пространством супермаркета представляет собой систему, направленную на автоматизацию работы с товарами, находящимися на полках супермаркета. Система предоставляет возможность анализа данных о продуктах, находящихся на полках и поступающих на них, в результате чего выдаются конкретные параметры оптимального количества товаров, необходимых для наиболее прибыльной работы супермаркета, а также границы количества товаров, за пределами которых система оповещает о возможных потерях прибыли супермаркета в случае недостатка количества товаров на полках либо его избытке. Кроме того, система дает возможность оповещать об истекшем сроке годности. Данная система также позволяет хранить статистические данные о работе супермаркета в каждый конкретный период, на основе чего она может выдавать рекомендуемые варианты расположения и количества товаров в определенные дни.

Разработка и внедрение данной системы позволит:

- Исключить или уменьшить человеческий фактор при заполнении и последующем наблюдении за полками;
- Автоматизировать процесс расчета оптимального количества товара на полках, а также процесс расчета граничных значений количества товаров;
- Выполнять расчет необходимых значений количества товаров на определенный период времени;
- Автоматизировать процесс подсчета количества поступающих на полки и проданных товаров.

Целесообразность инвестиций в разработку, внедрение и использование автоматизированной системы заключается в сокращении расходов на анализ данных о товарах на полках супермаркета, а также в сокращении затрат на персонал в связи с автоматизацией процессов мерчендайзинга и наблюдения за товарами.

В результате разработки и внедрения данной автоматизированной системы появится возможность автоматизации процессов, связанных с

работой с полочным пространством супермаркета, а также процессов сбора и анализа данных о поступивших и проданных товарах.

#### 4.2 Расчет инвестиций в разработку автоматизированной системы

Расчет затрат на основную заработную плату команды разработчиков осуществляется исходя из состава и численности команды, размера месячной заработной платы каждого участника команды, а также трудоемкости работ, выполняемых при разработке программного средства отдельными исполнителями по формуле 4.1:

$$З_o = K_{\text{пр}} \sum_{i=1}^n З_{\text{чи}} \cdot t_i, \quad (4.1)$$

где  $K_{\text{пр}}$  – коэффициент премий (по данным предприятия или в диапазоне 1,5–2);

$n$  – категории исполнителей, занятых разработкой программного средства;

$З_{\text{чи}}$  – часовая заработная плата исполнителя  $i$ -й категории, р.;

$t_i$  – трудоёмкость работ, выполняемых исполнителем  $i$ -й категории, определяется исходя из сложности разработки программного обеспечения и объёма выполняемых им функций, ч.

Коэффициент премий берется равным 1,5. Часовая заработная плата каждого исполнителя определяется путём деления его месячной заработной платы (оклад плюс надбавки) на количество рабочих часов в месяце, то есть на 168 часов.

При расчете заработной платы берется среднемесячная заработная плата в Республике Беларусь для сотрудников различных категорий ИТ-отрасли, поэтому премия не рассчитывается [9]. Расчёт затрат на основную заработную плату приведен в таблице 4.1.

Дополнительная заработная плата разработчиков рассчитывается по формуле 4.2:

$$З_d = \frac{З_o \cdot H_d}{100}, \quad (4.2)$$

где  $H_d$  – норматив дополнительной заработной платы, 10 %.

Таблица 4.1. – Расчет затрат на основную заработную плату команды разработчиков

| Категория исполнителя                                    | Месячная заработная плата, р. | Часовая заработная плата, р. | Трудоемкость работ, ч | Итого, р. |
|--|-------------------------------|------------------------------|-----------------------|-----------|
| 1  | 2                             | 3                            | 4                     | 5         |
| Бизнес-аналитик  | 3476,79                       | 20,7                         | 52,66                 | 1090,06   |
| Системный архитектор                                     | 7726,2                        | 45,99                        | 130,52                | 6002,62   |
| Программист  | 4378,18                       | 26,06                        | 133,14                | 3469,63   |
| Тестировщик  | 2575,4                        | 15,33                        | 97,59                 | 1496,05   |
| Итого  | 18156,57                      | 108,08                       | 413,88                | 12058,36  |
| Всего затраты на основную заработную плату разработчиков |                               |                              |                       | 18087,54  |

Отчисления на социальные нужды рассчитываются по формуле 4.3:

$$P_{\text{соц}} = \frac{(Z_o + Z_d) \cdot H_{\text{соц}}}{100}, \quad (4.3)$$

где  $H_{\text{соц}}$  – ставка отчислений в ФСЗН и Белгосстрах, 34,6 %.

Сумма прочих расходов рассчитывается по формуле 4.4:

$$P_{\text{пр}} = \frac{Z_o \cdot H_{\text{пр}}}{100}, \quad (4.4)$$

где  $H_{\text{пр}}$  – норматив прочих расходов, 35 %.

Общая сумма затрат на разработку рассчитывается по формуле 4.5:

$$Z_p = Z_o + Z_d + P_{\text{соц}} + P_{\text{пр}}, \quad (4.5)$$

Плановая прибыль, включаемая в цену программного средства рассчитывается по формуле 4.6:

$$P_{\text{пс}} = \frac{Z_p \cdot P_{\text{пс}}}{100}, \quad (4.6)$$

где  $P_{\text{пс}}$  – рентабельность затрат на разработку программного средства, 30 %.

Отпускная цена программного средства рассчитывается по формуле 4.7:

$$C_{\text{пс}} = Z_p + \Pi_{\text{пс}}, \quad (4.7)$$

Формирование цены на основе затрат приведено в таблице 4.2.

Таблица 4.2. – Формирование цены программного средства на основе затрат

| Наименование статьи затрат                                  | Расчет по формуле (в таблице)                                  | Значение, р. |
|---|--|--------------|
| 1   | 2  | 3            |
| 1 Основная заработная плата разработчиков                   | См. табл. 4.1  | 18087,54     |
| 2 Дополнительная заработная плата разработчиков             | $Z_d = \frac{18087,54 \cdot 10}{100}$                          | 1808,75      |
| 3 Отчисления на социальные нужды                            | $P_{\text{соц}} = \frac{(18087,54 + 1808,75) \cdot 34,6}{100}$ | 6884,12      |
| 4 Прочие расходы  | $P_{\text{пр}} = \frac{18087,54 \cdot 35}{100}$                | 6330,64      |
| 5 Общая сумма затрат на разработку                          | $Z_p = 18087,54 + 1808,75 + 6884,12 + 6330,64$                 | 33111,05     |
| 6 Плановая прибыль, включаемая в цену программного средства | $\Pi_{\text{пс}} = \frac{33111,05 \cdot 30}{100}$              | 9933,32      |
| 7 Отпускная цена программного средства                      | $C_{\text{пс}} = 33111,05 + 9933,32$                           | 43044,37     |

#### 4.3 Расчет результата разработки и использования автоматизированной системы

Экономический эффект от разработки программного средства по индивидуальному заказу может быть рассчитан как для организации-разработчика, так и для организации-заказчика.

#### 4.3.1 Расчет экономического эффекта для организации-разработчика

Для организации-разработчика экономическим эффектом является прирост чистой прибыли, полученной от разработки и реализации программного средства заказчику, который рассчитывается по формуле 4.8:

$$\Delta\Pi_{\text{ч}} = \Pi_{\text{пс}} \cdot \left(1 - \frac{H_{\text{п}}}{100}\right), \quad (4.8)$$

где  $\Pi_{\text{пс}}$  – прибыль, включаемая в цену программного средства, р.;  
 $H_{\text{п}}$  – ставка налога на прибыль, 18 %.

Таким образом, экономический эффект для организации-разработчика составит:

$$\Delta\Pi_{\text{ч}} = 9933,32 \cdot \left(1 - \frac{18}{100}\right) = 8145,32 \text{ р.}$$

#### 4.3.2 Расчет экономического эффекта для организации-заказчика

Для организации-заказчика экономическим эффектом в результате использования разрабатываемой автоматизированной системы является прирост чистой прибыли, полученный за счет:

- экономии затрат на заработную плату с начислениями на заработную плату служащих в связи с сокращением их численности;
- экономии затрат на оплату труда и прочих затрат в результате снижения влияния человеческого фактора;
- снижения затрат на заработную плату с начислениями на заработную плату основных производственных рабочих.

Экономия на заработной плате и начислениях на заработную плату сотрудников за счет снижения трудоемкости работ рассчитывается по формуле 4.9:

$$\begin{aligned} \mathcal{E}_{\text{зп}} = K_{\text{пр}} \cdot (t_{\text{р}}^{\text{без пс}} - t_{\text{р}}^{\text{с пс}}) \cdot T_{\text{ч}} \cdot N_{\text{п}} \cdot \left(1 + \frac{H_{\text{д}}}{100}\right) \times \\ \times \left(1 + \frac{H_{\text{но}}}{100}\right), \end{aligned} \quad (4.9)$$

где  $K_{пр}$  – коэффициент премий, 1,3;  
 $t_p^{без\ пс}$  – трудоемкость выполнения работ сотрудниками до внедрения автоматизированной системы, ч;  
 $t_p^{с\ пс}$  – трудоемкость выполнения работ сотрудниками после внедрения автоматизированной системы, ч;  
 $T_ч$  – часовая тарифная ставка сотрудника, использующего автоматизированную систему, р.;  
 $N_{п}$  – плановый объем работ, выполняемых сотрудником;  
 $H_{д}$  – норматив дополнительной заработной платы, 15 %;  
 $H_{но}$  – ставка отчислений от заработной платы, включаемых в себестоимость, 34,6 %.

Автоматизированная система предполагает снижение трудозатрат для такой категории рабочего персонала, как фасовщики-комплектовщики. Поэтому, исходя из того, что в супермаркете работает три фасовщика-комплектовщика,  $t_p^{без\ пс} = 6048$  часов в год,  $t_p^{с\ пс} = 4536$  часов в год,  $T_ч = 5$  рублей,  $N_{п} = 1$ . Соответственно экономия составит:

$$\mathcal{E}_{зп} = 1,3 \cdot (6048 - 4536) \cdot 5 \cdot 1 \cdot \left(1 + \frac{15}{100}\right) \left(1 + \frac{34,6}{100}\right) = 15212,76 \text{ р.}$$

Экономия на заработной плате и начислениях на заработную плату в результате сокращения численности работников рассчитывается по формуле 4.10:

$$\mathcal{E}_{зп} = K_{пр} \cdot \sum_{i=1}^n \Delta \mathcal{C}_i \cdot \mathcal{Z}_i \cdot \left(1 + \frac{H_{д}}{100}\right) \left(1 + \frac{H_{соц}}{100}\right), \quad (4.10)$$

где  $K_{пр}$  – коэффициент премий, 1,3;  
 $n$  – категории работников, высвобождаемых в результате внедрения программного средства;  
 $\Delta \mathcal{C}_i$  – численность работников  $i$ -й категории, высвобожденных после внедрения программного средства, чел.;  
 $\mathcal{Z}_i$  – годовая заработная плата высвобожденных работников  $i$ -й категории после внедрения программного средства, р.;  
 $H_{д}$  – норматив дополнительной заработной платы, 15 %;



$H_{\text{соц}}$  – норматив отчислений от заработной платы в соответствии с законодательством, 34,6 %.

С учетом вводимых новшеств, автоматизированная система позволит сэкономить на заработной плате при увольнении мерчендайзера и контролера торгового зала. Таким образом, если после введения автоматизированной системы годовая заработная плата мерчендайзеров будет составлять 11400 рублей, а контролера торгового зала – 6600 рублей, экономия составит:

$$\mathcal{E}_{\text{зн}} = 1,3 \cdot (1 \cdot 11400 + 1 \cdot 6600) \cdot \left(1 + \frac{15}{100}\right) \left(1 + \frac{35}{100}\right) = 36328,5 \text{ р.}$$

Соответственно экономия на текущих затратах составит:

$$\mathcal{E}_{\text{тек}} = 15212,76 + 36328,5 = 51541,26 \text{ р.}$$

Прирост чистой прибыли рассчитывается по формуле 4.11:

$$\Delta\Pi_{\text{ч}} = (\mathcal{E}_{\text{тек}} - \Delta\mathcal{Z}_{\text{тек}}^{\text{пс}}) \cdot \left(1 - \frac{H_{\text{п}}}{100}\right), \quad (4.11)$$

где  $\Delta\mathcal{Z}_{\text{тек}}^{\text{пс}}$  – прирост текущих затрат, связанных с использованием автоматизированной системы, р.

Для работы автоматизированной системы необходимо наличие интернета. Для этого производится подключение к интернет-провайдеру, услуги которого будут стоить 462 рубля в год. Кроме того, требуется своевременное обновление и решение проблем, возникающих в процессе работы системы. На поддержку и сопровождение будут выделяться средства в размере 40 % от итоговой стоимости системы, что составляет 17217,75 рубля. Затраты, связанные с использованием автоматизированной системы, составят:

$$\Delta\mathcal{Z}_{\text{тек}}^{\text{пс}} = 462 + 17217,75 = 17679,75 \text{ р.}$$

Следовательно, прирост чистой прибыли составит:

$$\Delta\Pi_{\text{ч}} = (51541,26 - 17679,75) \cdot \left(1 - \frac{18}{100}\right) = 27766,44 \text{ р.}$$

#### **4.4 Расчет показателей экономической эффективности разработки и использования автоматизированной системы**

Экономическая эффективность может быть рассчитана как для организации-разработчика, так и для организации-заказчика.

##### **4.4.1 Расчет показателей экономической эффективности разработки и использования автоматизированной системы для организации-разработчика**

Для организации-разработчика автоматизированной системы оценка экономической эффективности разработки осуществляется с помощью расчета простой нормы прибыли по формуле 4.12:

$$P_{\text{и}} = \frac{\Delta\Pi_{\text{ч}}}{З_{\text{р}}} \cdot 100 \%, \quad (4.12)$$

где  $\Delta\Pi_{\text{ч}}$  – прирост чистой прибыли, полученной от разработки автоматизированной системы организацией-разработчиком по индивидуальному заказу, р.;

$З_{\text{р}}$  – затраты на разработку автоматизированной системы организацией-разработчиком, р.

Таким образом, экономическая эффективность составит:

$$P_{\text{и}} = \frac{8145,32}{33111,05} \cdot 100 = 24,6 \%,$$

##### **4.4.2 Расчет показателей экономической эффективности разработки и использования автоматизированной системы для организации-заказчика**

Для организации-заказчика автоматизированной системы рассчитывается несколько показателей экономической эффективности, так как сумма инвестиций больше суммы годового экономического эффекта.

Прежде всего, для приведения доходов и затрат к настоящему моменту времени определяется коэффициент дисконтирования, который можно найти по формуле 4.13:

$$\alpha_t = \frac{1}{(1 + E_n)^{t-t_p}}, \quad (4.13)$$

где  $E_n$  – требуемая норма дисконта, которая по своей природе соответствует норме прибыли, устанавливаемой инвестором в качестве критерия рентабельности инвестиций, доли единицы;  
 $t$  – порядковый номер года, доходы и затраты которого приводятся к расчетному году;  
 $t_p$  – расчетный год, к которому приводятся доходы и инвестиционные затраты ( $t_p = 1$ ).

Норма дисконта принимается равной 0,09. Расчетный период составит 4 года. Таким образом, коэффициенты дисконтирования за каждый год составят:

$$\alpha_1 = \frac{1}{(1 + 0,09)^{1-1}} = 1$$

$$\alpha_2 = \frac{1}{(1 + 0,09)^{2-1}} = 0,92$$

$$\alpha_3 = \frac{1}{(1 + 0,09)^{3-1}} = 0,84$$

$$\alpha_4 = \frac{1}{(1 + 0,09)^{4-1}} = 0,77$$

Также надо учесть, что в первый год экономический эффект будет значительно меньше планируемого, так как в течение этого года осуществляется разработка системы. Для того, чтобы учесть этот факт, необходимо выяснить, сколько времени будет затрачено на разработку системы. В данном случае необходимо отталкиваться от наибольшего количества часов, затрачиваемых на разработку. Проанализировав таблицу 4.1 можно увидеть, что больше всего трудозатраты у программиста: он затратит на разработку 133,14 часов. Так как при работе над системой программист

отвечает не за все задачи, к данному времени необходимо добавить трудозатраты на такие этапы разработки, как планирование, анализ разрабатываемой системы, а также анализ и решение возникших при разработке проблем. Учитывая, что программист может работать над системой параллельно с определенными этапами, выявленные трудозатраты составят 35 часов. Поэтому общие трудозатраты на систему составят 168,14 часов, что равно одному рабочему году. Таким образом, выясняется, что в первый рассчитываемый год экономический эффект будет составлять 0 рублей.

Чистый дисконтированный доход – сумма дисконтированных значений потока платежей от проекта, приведенных к настоящему моменту времени. Он рассчитывается по формуле 4.14:

$$\text{ЧДД}(NPV) = \sum_{t=1}^n P_t \cdot \alpha_t - \sum_{t=1}^n Z_t \cdot \alpha_t, \quad (4.14)$$

где  $P_t$  – экономический эффект в году  $t$ , р.;

$Z_t$  – затраты (инвестиции) в году  $t$ , р.;

$\alpha_t$  – коэффициент дисконтирования, рассчитанный для года  $t$ ;

$n$  – расчетный период, количество лет.

Расчет чистого дисконтированного дохода за расчетный период приведен в таблице 4.3.

Таблица 4.3 – Расчёт эффективности инвестиций в разработку автоматизированной системы по индивидуальному заказу и его использование

| Показатель  | Значение по годам расчетного периода |          |          |          |
|---|--------------------------------------|----------|----------|----------|
|   | 1                                    | 2        | 3        | 4        |
| <b>Результат</b>  |                                      |          |          |          |
| 1. Прирост чистой прибыли, р.                             | 0                                    | 27766,44 | 27766,44 | 27766,44 |
| 2. Дисконтированный результат, р.                         | 0                                    | 25545,13 | 23323,81 | 21380,16 |
| <b>Затраты</b>  |                                      |          |          |          |
| 3. Инвестиции в разработку автоматизированной системы, р. | 33111,05                             | -        | -        | -        |
| 4. Дисконтированные инвестиции, р.                        | 33111,05                             | -        | -        | -        |
| 5. Чистый дисконтированный доход по годам, р.             | -33111,05                            | 25545,13 | 23323,81 | 21380,16 |
| 6. Чистый дисконтированный доход нарастающим итогом, р.   | -33111,05                            | -7565,92 | 15757,89 | 37138,05 |

Продолжение таблицы 4.3

|  |   |      |      |      |
|--|---|------|------|------|
| Коэффициент дисконтирования,<br>доли единицы | 1 | 0,92 | 0,84 | 0,77 |
|--|---|------|------|------|

Дисконтированный срок окупаемости – период возврата денежных средств с учетом временной стоимости денег. В отличие от срока окупаемости, данная величина приводит будущие денежные поступления к настоящему времени. Дисконтированный срок окупаемости будет равен моменту, когда суммарный дисконтированный результат превысит дисконтированную сумму инвестиций.

В данном случае дисконтированный эффект нарастающим итогом превысит дисконтированные инвестиции на третий год, поэтому дисконтированный срок окупаемости составит три года.

Рентабельность инвестиций без учета фактора времени рассчитывается по формуле 4.15:

$$P_{\text{и}} = \frac{\Pi_{\text{чср}}}{\sum_{t=1}^n Z_t} \cdot 100 \%, \quad (4.15)$$

где  $\Pi_{\text{чср}}$  – среднегодовая чистая прибыль;

Следовательно, рентабельность для организации-заказчика составит:

$$P_{\text{и}} = \frac{20824,83}{33111,05} \cdot 100 = 62,89 \%$$

Срок окупаемости инвестиций определяется по формуле 4.16:

$$T_{\text{ок}} = \frac{\sum_{t=1}^n Z_t}{P_{\text{ср}}}, \quad (4.16)$$

где  $P_{\text{ср}}$  – среднегодовая сумма результата (экономического эффекта), р.

Инвестиции вносятся в первый год разработки, то есть  $Z_1 = 33111,05$  р.,  $Z_2 = Z_3 = Z_4 = 0$  р. Среднегодовая сумма результата составляет 20824,83 р.

Таким образом, срок окупаемости инвестиций составит:

$$T_{\text{ок}} = \frac{33111,05 + 0 + 0 + 0}{20824,83} = 1,59 \text{ лет}$$

То есть автоматизированная система окупится за 1 год и 7 месяцев.

#### **4.5 Выводы об экономической эффективности и целесообразности инвестиций**

При разработке данной автоматизированной системы экономический эффект для организации-разработчика составит 8145,32 рубля. Кроме того, экономическая эффективность будет 24,6 %.

При внедрении автоматизированной системы для организации-заказчика расходы предприятия снизятся на 51541,26 рубля в год. Данная экономия связана с сокращением трудоемкости выполняемых работ персоналом и увольнением некоторых работников. Вложенные инвестиции окупятся чуть более чем за полтора года, а ежегодный прирост чистой прибыли составит 27766,44 рубля. Рентабельность инвестиций составит 62,89 %.

На основе данных показателей, можно сделать вывод, что внедрение автоматизированной системы полностью целесообразно.

## ЗАКЛЮЧЕНИЕ

В результате работы над дипломным проектом была разработана автоматизированная система управления полочным пространством супермаркета, которая автоматизирует взаимодействие персонала и менеджера с полками супермаркета. Также система дает возможность управлять продуктами, внесенными в базу данных, изменять их параметры, количество на полках и на складе. Кроме того, данная система имеет возможность автоматизировать процесс создания заданий и анализировать поступающие в процессе работы данные. Бизнес-логика создана в соответствии с обозначенными требованиями, реализованы основные функции, планировавшиеся в начале работы над проектом.

При работе над дипломным проектом были выделены аналоги данной системы, проведен их анализ и выявлены некоторые особенности, благодаря чему удалось сформировать возможные функции, которыми будет обладать разработанная система. Некоторые возможности системы разработаны с целью дальнейшего развития, поэтому могут иметь не полный на первый взгляд функционал.

Также созданы дополнительные функции, которые направлены на взаимодействие системы с аппаратным обеспечением супермаркета. Данная связь планируется к разработке под конкретный супермаркет, так как аппаратное обеспечение может отличаться в зависимости от торгового объекта, на котором планируется внедрение системы.

При проведении технико-экономического обоснования разработки и использования автоматизированной системы были выявлены основные затраты и ожидаемый доход от внедрения.

Дипломный проект выполнен и оформлен в соответствии с действующими стандартом предприятия и государственными стандартами [10].

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- [1] Виды и типы магазинов розничной торговли [Электронный ресурс]. – Режим доступа : <https://litstile.ru/index.php?ht=1674>.
- [2] Супермаркет [Электронный ресурс]. – Режим доступа : <https://znaytovar.ru/s/Supermarket.html>.
- [3] Что такое гипермаркет [Электронный ресурс]. – Режим доступа : <https://znaytovar.ru/s/chto-takoe-gipermarket.html>.
- [4] Автоматизируем магазин продуктов [Электронный ресурс]. – Режим доступа : <https://cloudshop.ru/content/avtomatiziruem-magazin-produktov>.
- [5] Терминал сбора данных [Электронный ресурс]. – Режим доступа : <https://shtrih-m-spb.ru/catalog/tsd/>.
- [6] Классификация сканеров штрих-кода [Электронный ресурс]. – Режим доступа : <https://posmagazin.ru/articles/klassifikatsiya-skanerov-shtrikh-koda/>.
- [7] Система управления полочным пространством магазина, автоматизация управления выкладкой [Электронный ресурс] – Режим доступа : <https://abmcloud.com/abm-soft/shelf/>.
- [8] *RS.ShelfSpace*: система управления полочным пространством магазина, автоматизация мерчандайзинга, контроль выкладки товаров [Электронный ресурс] – Режим доступа : <https://shelfspace.ru/>.
- [9] Зарплата в ИТ [Электронный ресурс] – Режим доступа : <https://salaries.dev.by/>.
- [10] СТП 01-2017. Стандарт предприятия. Дипломные проекты (работы). Общие требования. - Минск: БГУИР, 2017. - 169 с.



## ПРИЛОЖЕНИЕ А

### (обязательное)

### Программный код модуля анализа данных

```
from .. import models
from . import helpers
import datetime
import math
from _datetime import timedelta

def getFullStats():
    storeShelves = models.Store.objects.all()
    fullStatsRaw = models.Statistics.objects.order_by('day')

    years = list(
        map(
            lambda singleYearRaw: singleYearRaw.strftime('%Y'),
            models.Statistics.objects.all().dates('day', 'year')
        )
    )

    fullStatsForEachShelf = []
    for storeShelf in storeShelves:
        soldInYear = {}
        incomeInYear = {}
        failuresInYear = {}
        for year in years:
            soldInYear[year] = []
            incomeInYear[year] = []
            failuresInYear[year] = 0

        for dailyStats in fullStatsRaw:
            if storeShelf == dailyStats.shelf:
                year = dailyStats.statYear
                soldInYear[year].append(
                    [dailyStats.day.strftime(
                        '%m/%d'), dailyStats.sold_count]
                )
            if dailyStats.stock:
                incomeInYear[year].append(
                    [dailyStats.day.strftime(
                        '%m/%d'), dailyStats.sold_count * dailyStats.stock.stock_price]
                )
            else:
                incomeInYear[year].append(
                    [dailyStats.day.strftime(
                        '%m/%d'), dailyStats.sold_count * dailyStats.price_that_day]
                )
```

```

        failuresInYear[year] += dailyStats.failures_count
    availableStats = {}
    if soldInYear:
        availableStats["soldEveryDay"] = soldInYear
    if incomeInYear:
        availableStats["incomeEveryDay"] = incomeInYear
    if failuresInYear:
        availableStats["failuresInYear"] = failuresInYear
    fullStatsForEachShelf.append(
        [storeShelf, availableStats]
    )
    return fullStatsForEachShelf

```

```

def handleNewData(payload):
    storeElemId = payload['storeElem_id']
    result = helpers.getEmptyResultObject()

```

```

    storeElem = models.Store.objects.get(id=storeElemId)

```

```

    day = payload['day']
    try:
        day = datetime.datetime.strptime(day, "%Y/%m/%d")
    except ValueError:
        day = None
        result['failure'] = "неправильный формат даты"
    return result

```

```

    sold_count = payload['sold_count']
    try:
        sold_count = int(sold_count)
    except ValueError:
        sold_count = None
        result['failure'] = "количество проданных продуктов должно быть целым
числом"
    return result

```

```

    stock_id = payload['stock_id']
    stock = None
    if stock_id != "":
        try:
            stock = models.Stock.objects.get(id=stock_id)
        except models.Stock.DoesNotExist:
            result['failure'] = "акция не найдена"
        return result

```

```

    failures_count = payload['failures_count']
    try:
        failures_count = int(failures_count)
    except ValueError:
        result['failure'] = "количество ошибок должно быть целым числом"

```

```

        return result

    price_that_day = payload['price_that_day']
    try:
        price_that_day = float(price_that_day)
    except ValueError:
        result['failure'] = "цена в указанный день должна быть целым или
дробным числом"
        return result

    statsElem = models.Statistics.objects.create(
        day=day,
        shelf=storeElem,
        sold_count=sold_count,
        stock=stock,
        failures_count=failures_count,
        price_that_day=price_that_day
    )
    result['success'] = statsElem.toJSON()
    return result


def calculateLastYearStats(currentDate, shelf, min_products_count,
max_products_count, years_to_count, days_ago):
    stats_count = 0
    min_that_year = 0
    max_that_year = 0
    days_count = 10
    for i in range(1, days_count + 1):
        day = currentDate - timedelta(days=days_ago+i)
        try:
            stat = models.Statistics.objects.get(shelf=shelf, day=day)
            stats_count += 1
            if max_that_year < stat.sold_count:
                max_that_year = stat.sold_count
            if min_that_year > stat.sold_count:
                min_that_year = stat.sold_count
        except models.Statistics.DoesNotExist:
            pass

    if min_that_year < min_products_count:
        min_that_year = min_products_count
    if max_that_year > max_products_count:
        max_that_year = max_products_count

    if stats_count != 0:
        years_to_count += 1

    return [min_that_year, max_that_year, years_to_count]

```

```

def getStatsForShelf(shelf):
    currentDate = datetime.date.today()
    # 1. Essential borders

    min_products_count = math.floor(shelf.width / shelf.product.width)
    while min_products_count * shelf.product.weight >= 0.75 *
shelf.carrying_capacity:
    min_products_count -= 1
    max_products_count = 0
    product_S = shelf.product.width * shelf.product.length
    product_V = product_S * shelf.product.height
    shelf_S = shelf.width * shelf.length
    shelf_V = shelf_S * shelf.height
    if shelf.product.stackable:
        max_products_count = math.floor(shelf_V / product_V)
    else:
        max_products_count = math.floor(shelf_S / product_S)
    while max_products_count * shelf.product.weight >= 0.75 *
shelf.carrying_capacity:
    max_products_count -= 1
    if min_products_count > max_products_count:
        min_products_count = max_products_count

    products_count_average = (max_products_count + min_products_count) / 2

    years_to_count = 1

    # 2. Last few days borders

    relevance = 0.5
    stats_count = 0
    min_for_day_sum = 0
    max_for_day_sum = 0
    days_count = 10
    for i in range(1, days_count + 1):
        day = currentDate - timedelta(days=i)
        try:
            stat = models.Statistics.objects.get(shelf=shelf, day=day)
            stats_count += 1
            disp = stat.sold_count - stat.sold_count * relevance
            max_for_day_sum += stat.sold_count + disp
            min_for_day_sum += stat.sold_count - disp
        except models.Statistics.DoesNotExist:
            pass
        relevance = (2**(days_count - i - 1))/(2**days_count)
    min_recommended = 0
    max_recommended = 0
    if stats_count != 0:
        min_recommended = math.floor(min_for_day_sum / stats_count)
        max_recommended = math.ceil(max_for_day_sum / stats_count)
    years_to_count += 1

```

```

    if min_recommended < min_products_count:
        min_recommended = min_products_count
    if max_recommended > max_products_count:
        max_recommended = max_products_count

# 3. Last year borders

    calculatedStats = calculateLastYearStats(
        currentDate, shelf, min_products_count, max_products_count,
        years_to_count, 365
    )
    min_year_ago = calculatedStats[0]
    max_year_ago = calculatedStats[1]
    years_to_count = calculatedStats[2]

# 4. Two years ago borders

    calculatedStats = calculateLastYearStats(
        currentDate, shelf, min_products_count, max_products_count,
        years_to_count, 365 * 2
    )
    min_two_years_ago = calculatedStats[0]
    max_two_years_ago = calculatedStats[1]
    years_to_count = calculatedStats[2]

# 5. Result

    min_final = math.ceil((min_products_count + min_recommended +
        min_year_ago + min_two_years_ago) / years_to_count)
    max_final = math.floor((max_products_count + max_recommended +
        max_year_ago + max_two_years_ago) / years_to_count)
    avg_recommended = math.ceil(
        ((min_final + max_final)/2 + products_count_average)/2)
    return [min_final, max_final, avg_recommended]

def calculateNextDay(payload):
    result = helpers.getEmptyResultObject()

    statsJSON = "{"
    storeElems = models.Store.objects.all()
    for shelf in storeElems:
        shelfStats = getStatsForShelf(shelf)
        min_final = shelfStats[0]
        max_final = shelfStats[1]
        avg_recommended = shelfStats[2]
        statsJSON += '"' + str(shelf.id) + "":{"min_products':" + str(min_final) + ",
"max_products':" + str(
            max_final) + ", "avg_recommended':" + str(avg_recommended) + "},"
        result['success'] = statsJSON[:-1] + '}'
    return result

```

Ведомость в отдельном доке